消息队列kafka（消息中间件）

问题

```
1. 幂等性
2. 在producer端设置retries=MAX（很大很大很大的一个值，无限次重试的意思）：这个是要求一旦写入失败，就无限重试，卡在这里了，如何处理
3.
```

MQ的作用：解耦，限流，削峰

随笔

一个partition只能被一个消费者消费，一个消费者可以消费多个partition
生产者将消息分发到kafka

1. 消息生产者：即：Producer，是消息的产生的源头，负责生成消息并发送到Kafka服务器上。
2. 消息消费者：即：Consumer，是消息的使用方，负责消费Kafka服务器上的消息。
3. 主题：即：Topic，由用户定义并配置在Kafka服务器，用于建立生产者和消息者之间的订阅关系：生产者发送消息到指定的Topic下，消息者从这个Topic下消费消息。
4. 消息分区：即：Partition，一个Topic下面会分为很多分区，例如：“kafka-test”这个Topic下可以分为6个分区，分别由两台服务器提供，那么通常可以配置为让每台服务器提供3个分区，假如服务器ID分别为0、1，则所有的分区为0-0、0-1、0-2和1-0、1-1、1-2。Topic物理上的分组，一个 topic可以分为多个 partition，每个 partition 是一个有序的队列。partition中的每条消息都会被分配一个有序的id（offset）。
5. Broker：即Kafka的服务器，用户存储消息，Kafa集群中的一台或多台服务器统称为 broker。
6. 消费者分组：Group，用于归组同类消费者，在Kafka中，多个消费者可以共同消息一个Topic下的消息，每个消费者消费其中的部分消息，这些消费者就组成了一个分组，拥有同一个分组名称，通常也被称为消费者集群。
7. Offset：消息存储在Kafka的Broker上，消费者拉取消息数据的过程中需要知道消息在文件中的偏移量，这个偏移量就是所谓的Offset。

Message在Broker中通Log追加的方式进行持久化存储。并进行分区（patitions）。为了减少磁盘写入的次数,broker会将消息暂时buffer起来,当消息的个数(或尺寸)达到一定阀值时,再flush到磁盘,这样减少了磁盘IO调用的次数。
Broker没有副本机制，一旦broker宕机，该broker的消息将都不可用。Message消息是有多份的。
Broker不保存订阅者的状态，由订阅者自己保存。
无状态导致消息的删除成为难题（可能删除的消息正在被订阅），kafka采用基于时间的SLA(服务水平保证)，消息保存一定时间（通常为7天）后会被删除。

partition中的每条Message包含了以下三个属性：
offset　　即：消息唯一标识:对应类型：long
MessageSize 对应类型：int32
data　　是message的具体内容。

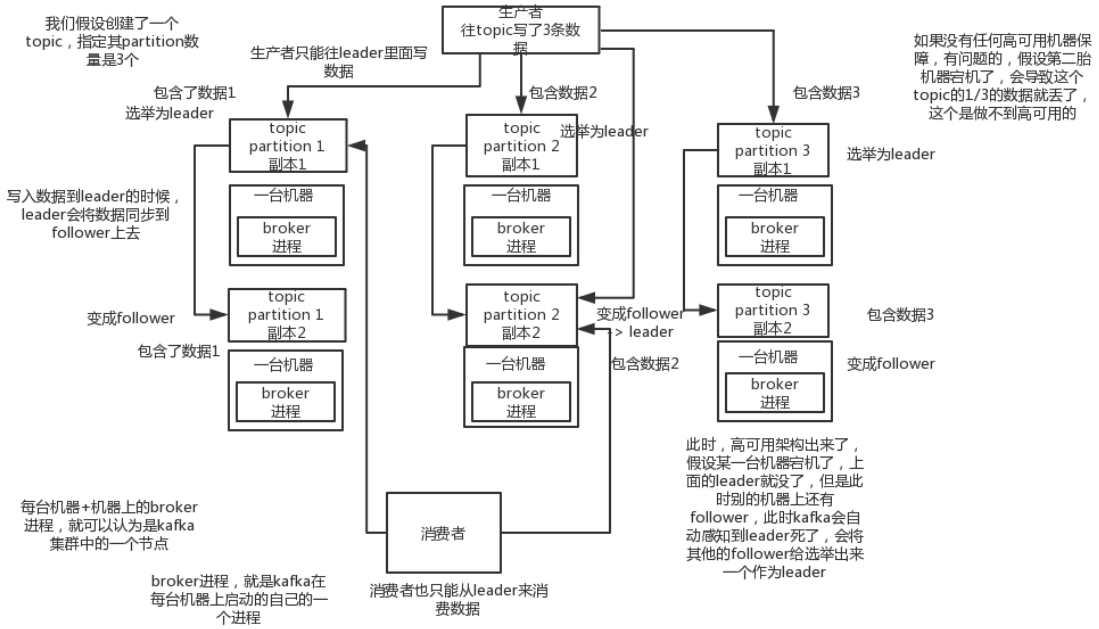
一个Topic中的每个partions，只会被一个“订阅者”中的一个consumer消费，不过一个 consumer可以消费多个partitions中的消息（消费者数据小于Partions的数量时）

一个Topic可以认为是一类消息，每个topic将被分成多partition(区),每个partition在存储层面是append log文件。任何发布到此partition的消息都会被直接追加到log文件的尾部，每条消息在文件中的位置称为offset（偏移量），partition是以文件的形式存储在文件系统中。

partitions的设计目的有多个。最根本原因是kafka基于文件存储。通过分区,可以将日志内容分散到多个server 上,来避免文件尺寸达到单机磁盘的上限,每个partiton都会被当前server(kafka实例)保存;可以将一个topic 切分多任意多个partitions来保存消息。此外越多的partitions 意味着可以容纳更多的consumer,有效提升并发消费的能力。

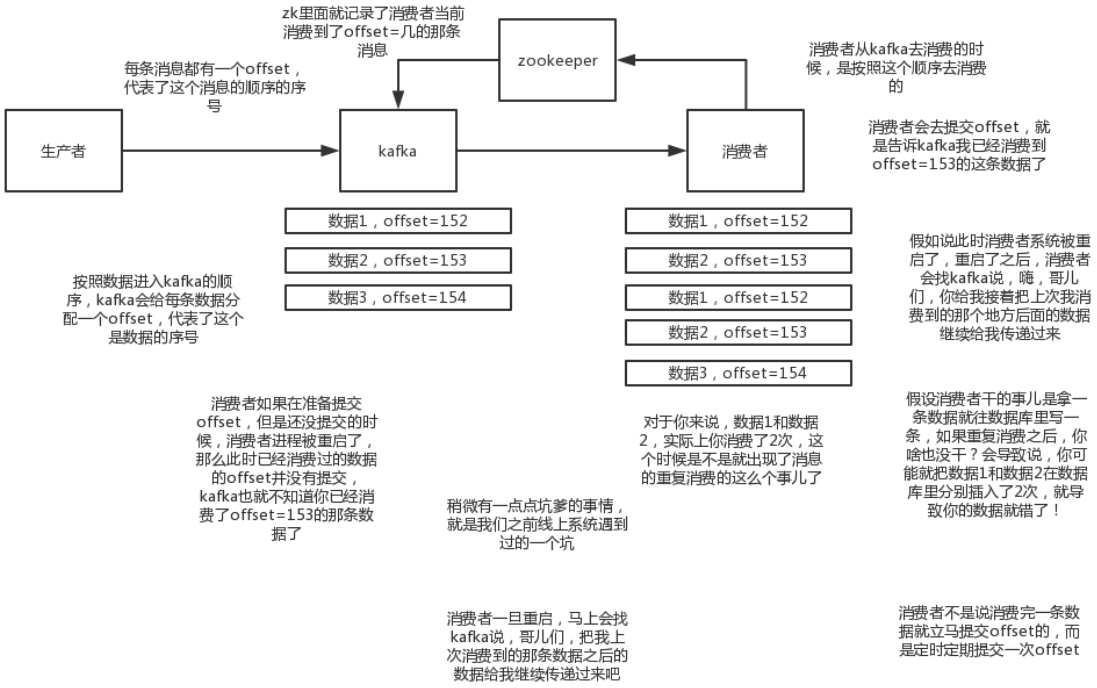
高可用

> 概念



生产者将消息写在了分别位于三台服务器的topic上，每台服务器上都会启动一个broker进程，其中会有多个partition副本，有一个被选举为leader，负责读写入消息，其他多个为follower，只可以读，用来备份消息，当leader宕机，会在follower中推举出一个作为新的leader，进行写入消息的操作。消费者只能从leader中消费数据

保证消息不被重复消费（幂等性）

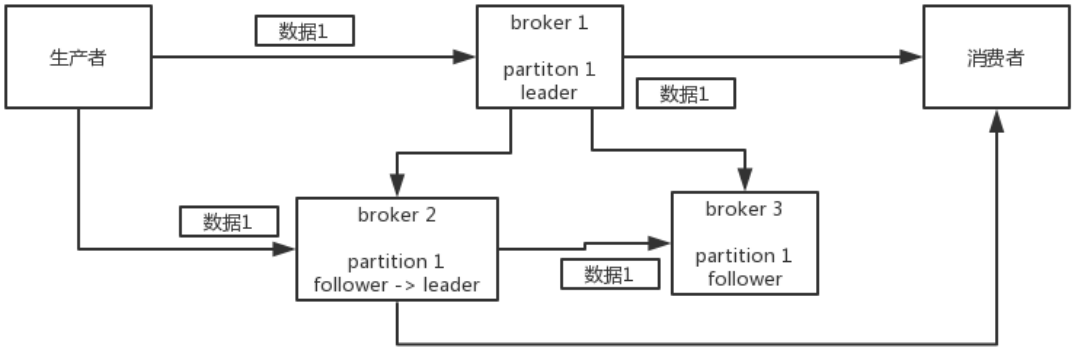


生产者向kafka写入消息时会提交一个offset，这个代表每个消息的顺序消费者按照offset进行消费消息，消费时会给zookeeper提交这个offset（是定期提交的），代表这个消息已经消费过了，消费者重启后在去消费时会继续向后消费。

- 当这次offset还没有提交时，消费者重启，offset没有提交，下次消费就会出现重复消费，数据就会有问题。

处理： 拿到消费的消息后，先暂存在一个地方，如redis中下次拿到消费的消息时，先判断之前有没有消费过，如果有，则不对数据库进行操作，保证数据的幂等性。

保证消息的可靠性传输，数据不丢失



1. 数据丢失的原因

① 消费者将数据丢失，消费者拿到消息后，想zookeeper提交了offset后，还没来得及对消息进行操作时，消费者宕机

② kafka将数据丢失，生产者发送了消息1，leader拿到消息后，还没来得及同步到follower上时leader发生宕机，消息1丢失

2. 处理

①取消自动提交offset，等数据处理完后再进行提交offset

②生产环境也遇到过，我们也是，之前kafka的leader机器宕机了，将follower切换为leader之后，就会发现说这个数据就丢了

所以此时一般是要求起码设置如下4个参数：

给这个topic设置replication.factor参数：这个值必须大于1，要求每个partition必须有至少2个副本

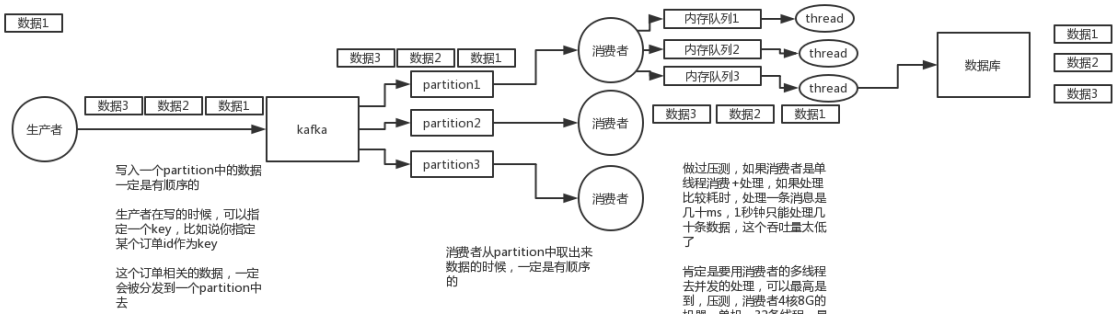
在kafka服务端设置min.insync.replicas参数：这个值必须大于1，这个是要求一个leader至少感知到有至少一个follower还跟自己保持联系，没掉队，这样才能确保leader挂了还有一个follower吧

在producer端设置acks=all：这个是要求每条数据，必须是写入所有replica之后，才能认为是写成功了

在producer端设置retries=MAX（很大很大很大的一个值，无限次重试的意思）：这个是要求一旦写入失败，就无限重试，卡在这里了

我们生产环境就是按照上述要求配置的，这样配置之后，至少在kafka broker端就可以保证在leader所在broker发生故障，进行leader切换时，数据不会丢失

kafka消息顺序错乱



当消费者开启多线程对消息进行处理时，数据顺序可能发生错乱，
使用内存队列，将需要有顺序的数据分配同一个key，将同一个key的数据分配到统一个线程中，，在存入到数据库中时就能保证数据的顺序

如何解决消息队列的延时以及过期失效问题？消息队列满了以后该怎么处理？有几百万消息持续积压几小时，说说怎么解决？

- 10_

当消费者故障时，消息发生了积压