Kafka 简介

Apache Kafka 是一个分布式发布-订阅消息系统。是大数据领域消息队列中唯一的王者。最初由 linkedin 公司使用 scala 语言开发，在2010年贡献给了Apache基金会并成为顶级开源项目。至今已有十余年，仍然是大数据领域不可或缺的并且是越来越重要的一个组件。

Kafka 适合离线和在线消息，消息保留在磁盘上，并在集群内复制以防止数据丢失。kafka构建在zookeeper同步服务之上。它与 Flink 和 Spark 有非常好的集成，应用于实时流式数据分析。

Kafka特点：

可靠性：具有副本及容错机制。

可扩展性：kafka无需停机即可扩展节点及节点上线。

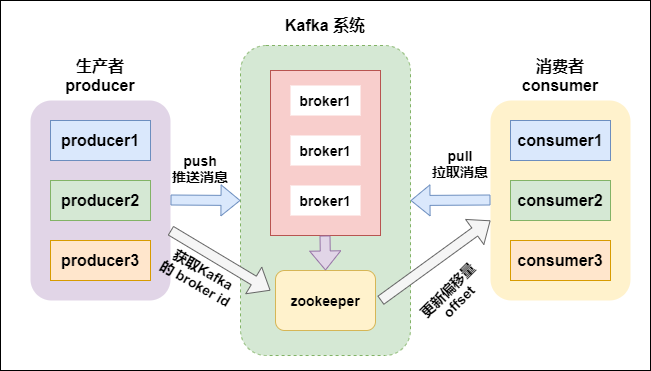
持久性：数据存储到磁盘上，持久性保存。

性能：kafka具有高吞吐量。达到TB级的数据，也有非常稳定的性能。

速度快：顺序写入和零拷贝技术使得kafka延迟控制在毫秒级。

Kafka 底层原理

先看下 Kafka 系统的架构



kafka支持消息持久化，消费端是主动拉取数据，消费状态和订阅关系由客户端负责维护，消息消费完后，不会立即删除，会保留历史消息。因此支持多订阅时，消息只会存储一份就可以。

broker：kafka集群中包含一个或者多个服务实例（节点），这种服务实例被称为broker（一个broker就是一个节点/一个服务器）；

topic：每条发布到kafka集群的消息都属于某个类别，这个类别就叫做topic；

partition：partition是一个物理上的概念，每个topic包含一个或者多个partition；

segment：一个partition当中存在多个segment文件段，每个segment分为两部分，.log文件和 .index 文件，其中 .index 文件是索引文件，主要用于快速查询， .log 文件当中数据的偏移量位置；

producer：消息的生产者，负责发布消息到 kafka 的 broker 中；

consumer：消息的消费者，向 kafka 的 broker 中读取消息的客户端；

consumer group：消费者组，每一个 consumer 属于一个特定的 consumer group（可以为每个consumer指定 groupName）；

.log：存放数据文件；

.index：存放.log文件的索引数据。

Kafka 主要组件

1. producer（生产者）

producer主要是用于生产消息，是kafka当中的消息生产者，生产的消息通过topic进行归类，保存到kafka的broker里面去。

2. topic（主题）

kafka将消息以topic为单位进行归类；

topic特指kafka处理的消息源（feeds of messages）的不同分类；

topic是一种分类或者发布的一些列记录的名义上的名字。kafka主题始终是支持多用户订阅的；也就是说，一 个主题可以有零个，一个或者多个消费者订阅写入的数据；

在kafka集群中，可以有无数的主题；

生产者和消费者消费数据一般以主题为单位。更细粒度可以到分区级别。

3. partition（分区）

kafka当中，topic是消息的归类，一个topic可以有多个分区（partition），每个分区保存部分topic的数据，所有的partition当中的数据全部合并起来，就是一个topic当中的所有的数据。

一个broker服务下，可以创建多个分区，broker数与分区数没有关系；

在kafka中，每一个分区会有一个编号：编号从0开始。

每一个分区内的数据是有序的，但全局的数据不能保证是有序的。（有序是指生产什么样顺序，消费时也是什么样的顺序）

4. consumer（消费者）

consumer是kafka当中的消费者，主要用于消费kafka当中的数据，消费者一定是归属于某个消费组中的。

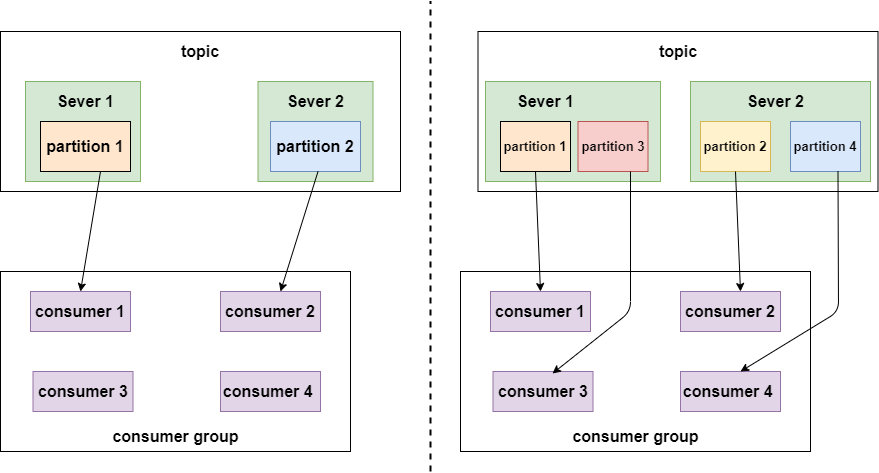
5. consumer group（消费者组）

消费者组由一个或者多个消费者组成，同一个组中的消费者对于同一条消息只消费一次。

每个消费者都属于某个消费者组，如果不指定，那么所有的消费者都属于默认的组。

每个消费者组都有一个ID，即group ID。组内的所有消费者协调在一起来消费一个订阅主题( topic)的所有分区(partition)。当然，每个分区只能由同一个消费组内的一个消费者(consumer)来消费，可以由不同的消费组来消费。

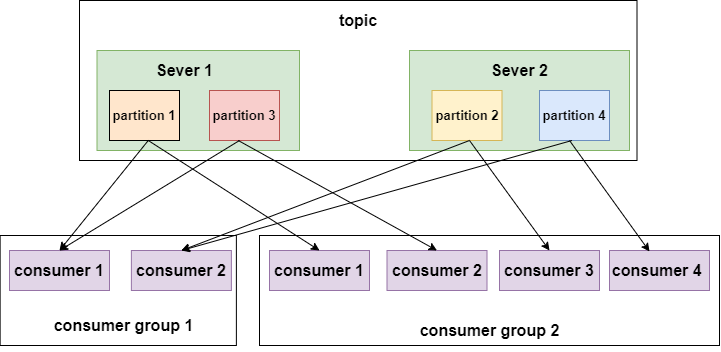
partition数量决定了每个consumer group中并发消费者的最大数量。如下图：



如上面左图所示，如果只有两个分区，即使一个组内的消费者有4个，也会有两个空闲的。

如上面右图所示，有4个分区，每个消费者消费一个分区，并发量达到最大4。

在来看如下一幅图：



如上图所示，不同的消费者组消费同一个topic，这个topic有4个分区，分布在两个节点上。左边的 消费组1有两个消费者，每个消费者就要消费两个分区才能把消息完整的消费完，右边的 消费组2有四个消费者，每个消费者消费一个分区即可。

总结下kafka中分区与消费组的关系：

消费组： 由一个或者多个消费者组成，同一个组中的消费者对于同一条消息只消费一次。

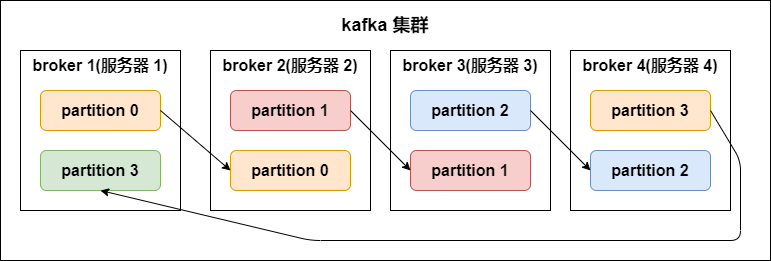
某一个主题下的分区数，对于消费该主题的同一个消费组下的消费者数量，应该小于等于该主题下的分区数。

如：某一个主题有4个分区，那么消费组中的消费者应该小于等于4，而且最好与分区数成整数倍 1 2 4 这样。同一个分区下的数据，在同一时刻，不能同一个消费组的不同消费者消费。

总结：分区数越多，同一时间可以有越多的消费者来进行消费，消费数据的速度就会越快，提高消费的性能。

6. partition replicas（分区副本）

kafka 中的分区副本如下图所示：



副本数（replication-factor）：控制消息保存在几个broker（服务器）上，一般情况下副本数等于broker的个数。

一个broker服务下，不可以创建多个副本因子。创建主题时，副本因子应该小于等于可用的broker数。

副本因子操作以分区为单位的。每个分区都有各自的主副本和从副本；

主副本叫做leader，从副本叫做 follower（在有多个副本的情况下，kafka会为同一个分区下的所有分区，设定角色关系：一个leader和N个 follower），处于同步状态的副本叫做in-sync-replicas(ISR);

follower通过拉的方式从leader同步数据。

消费者和生产者都是从leader读写数据，不与follower交互。

副本因子的作用：让kafka读取数据和写入数据时的可靠性。

副本因子是包含本身，同一个副本因子不能放在同一个broker中。

如果某一个分区有三个副本因子，就算其中一个挂掉，那么只会剩下的两个中，选择一个leader，但不会在其他的broker中，另启动一个副本（因为在另一台启动的话，存在数据传递，只要在机器之间有数据传递，就会长时间占用网络IO，kafka是一个高吞吐量的消息系统，这个情况不允许发生）所以不会在另一个broker中启动。

如果所有的副本都挂了，生产者如果生产数据到指定分区的话，将写入不成功。

lsr表示：当前可用的副本。

7. segment文件

一个partition当中由多个segment文件组成，每个segment文件，包含两部分，一个是 .log 文件，另外一个是 .index 文件，其中 .log 文件包含了我们发送的数据存储，.index 文件，记录的是我们.log文件的数据索引值，以便于我们加快数据的查询速度。

索引文件与数据文件的关系

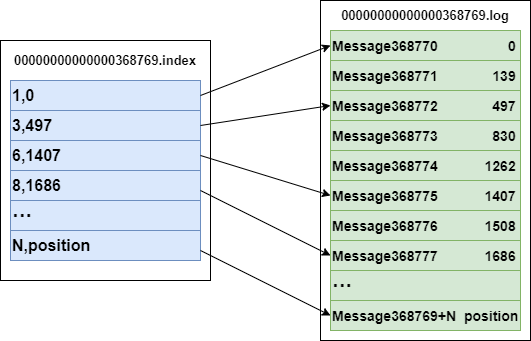
既然它们是一一对应成对出现，必然有关系。索引文件中元数据指向对应数据文件中message的物理偏移地址。

比如索引文件中 3,497 代表：数据文件中的第三个message，它的偏移地址为497。

再来看数据文件中，Message 368772表示：在全局partiton中是第368772个message。

注：segment index file 采取稀疏索引存储方式，减少索引文件大小，通过mmap（内存映射）可以直接内存操作，稀疏索引为数据文件的每个对应message设置一个元数据指针，它比稠密索引节省了更多的存储空间，但查找起来需要消耗更多的时间。

.index 与 .log 对应关系如下：



上图左半部分是索引文件，里面存储的是一对一对的key-value，其中key是消息在数据文件（对应的log文件）中的编号，比如“1,3,6,8……”，

分别表示在log文件中的第1条消息、第3条消息、第6条消息、第8条消息……

那么为什么在index文件中这些编号不是连续的呢？

这是因为index文件中并没有为数据文件中的每条消息都建立索引，而是采用了稀疏存储的方式，每隔一定字节的数据建立一条索引。

这样避免了索引文件占用过多的空间，从而可以将索引文件保留在内存中。

但缺点是没有建立索引的Message也不能一次定位到其在数据文件的位置，从而需要做一次顺序扫描，但是这次顺序扫描的范围就很小了。

value 代表的是在全局partiton中的第几个消息。

以索引文件中元数据 3,497 为例，其中3代表在右边log数据文件中从上到下第3个消息，

497表示该消息的物理偏移地址（位置）为497(也表示在全局partiton表示第497个消息-顺序写入特性)。

log日志目录及组成

kafka在我们指定的log.dir目录下，会创建一些文件夹；名字是 （主题名字-分区名） 所组成的文件夹。 在（主题名字-分区名）的目录下，会有两个文件存在，如下所示：

#索引文件

00000000000000000000.index

#日志内容

00000000000000000000.log

1

2

3

4

在目录下的文件，会根据log日志的大小进行切分，.log文件的大小为1G的时候，就会进行切分文件；如下：

-rw-r--r--. 1 root root 389k 1月 17 18:03 00000000000000000000.index

-rw-r--r--. 1 root root 1.0G 1月 17 18:03 00000000000000000000.log

-rw-r--r--. 1 root root 10M 1月 17 18:03 00000000000000077894.index

-rw-r--r--. 1 root root 127M 1月 17 18:03 00000000000000077894.log

1

2

3

4

在kafka的设计中，将offset值作为了文件名的一部分。

segment文件命名规则：partion全局的第一个segment从0开始，后续每个segment文件名为上一个全局 partion的最大offset（偏移message数）。数值最大为64位long大小，20位数字字符长度，没有数字就用 0 填充。

通过索引信息可以快速定位到message。通过index元数据全部映射到内存，可以避免segment File的IO磁盘操作；

通过索引文件稀疏存储，可以大幅降低index文件元数据占用空间大小。

稀疏索引：为了数据创建索引，但范围并不是为每一条创建，而是为某一个区间创建；

好处：就是可以减少索引值的数量。

不好的地方：找到索引区间之后，要得进行第二次处理。

8. message的物理结构

生产者发送到kafka的每条消息，都被kafka包装成了一个message

message 的物理结构如下图所示：



所以生产者发送给kafka的消息并不是直接存储起来，而是经过kafka的包装，每条消息都是上图这个结构，只有最后一个字段才是真正生产者发送的消息数据。

kafka中的数据不丢失机制

1. 生产者生产数据不丢失

发送消息方式

生产者发送给kafka数据，可以采用同步方式或异步方式

同步方式：

发送一批数据给kafka后，等待kafka返回结果：

生产者等待10s，如果broker没有给出ack响应，就认为失败。

生产者重试3次，如果还没有响应，就报错.

异步方式：

发送一批数据给kafka，只是提供一个回调函数：

先将数据保存在生产者端的buffer中。buffer大小是2万条 。

满足数据阈值或者数量阈值其中的一个条件就可以发送数据。

发送一批数据的大小是500条。

注：如果broker迟迟不给ack，而buffer又满了，开发者可以设置是否直接清空buffer中的数据。

ack机制（确认机制）

生产者数据发送出去，需要服务端返回一个确认码，即ack响应码；ack的响应有三个状态值0,1，-1

0：生产者只负责发送数据，不关心数据是否丢失，丢失的数据，需要再次发送

1：partition的leader收到数据，不管follow是否同步完数据，响应的状态码为1

-1：所有的从节点都收到数据，响应的状态码为-1

如果broker端一直不返回ack状态，producer永远不知道是否成功；producer可以设置一个超时时间10s，超过时间认为失败。

2. broker中数据不丢失

在broker中，保证数据不丢失主要是通过副本因子（冗余），防止数据丢失。

3. 消费者消费数据不丢失

在消费者消费数据的时候，只要每个消费者记录好offset值即可，就能保证数据不丢失。也就是需要我们自己维护偏移量(offset)，可保存在 Redis 中。