# 课程介绍

课程名称：

Kafka技术增强

注：学习本课程请先学习Kafka基础

课程目标：

通过本模块的学习，能够掌握Kafka的负载均衡、Producer生产数据、Kafka文件存储机制、Kafka自定义partition

课程大纲：

1. Kafka整体结构图
2. Consumer与topic关系
3. Kafka Producer消息分发
4. Consumer 的负载均衡
5. Kafka文件存储机制

# 课程内容

## 1、Kafka整体结构图

Kafka名词解释和工作方式

* Producer ：消息生产者，就是向kafka broker发消息的客户端。
* Consumer ：消息消费者，向kafka broker取消息的客户端
* Topic ：咋们可以理解为**一个队列**。
* Consumer Group （CG）：这是kafka用来实现一个topic消息的广播（发给所有的consumer）和单播（发给任意一个consumer）的手段。一个topic可以有多个CG。topic的消息会复制（不是真的复制，是概念上的）到所有的CG，但每个partion只会把消息发给该CG中的一个consumer。如果需要实现广播，只要每个consumer有一个独立的CG就可以了。要实现单播只要所有的consumer在同一个CG。用CG还可以将consumer进行自由的分组而不需要多次发送消息到不同的topic。
* Broker ：一台kafka服务器就是一个broker。一个集群由多个broker组成。一个broker可以容纳多个topic。
* Partition：为了实现扩展性，一个非常大的topic可以分布到多个broker（即服务器）上，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）**。**kafka只保证按一个partition中的顺序将消息发给consumer，不保证一个topic的整体（多个partition间）的顺序**。
* Offset：kafka的存储文件都是按照offset.kafka来命名，用offset做名字的好处是方便查找。例如你想找位于2049的位置，只要找到2048.kafka的文件即可。当然the first offset就是00000000000.kafka

## 2、Consumer与topic关系

本质上kafka只支持Topic；

* 每个group中可以有多个consumer，每个consumer属于一个consumer group；

通常情况下，**一个group中会包含多个consumer**，这样不仅可以提高topic中消息的并发消费能力，而且还能提高"故障容错"性，如果group中的某个consumer失效那么其消费的partitions将会有其他consumer自动接管。

* 对于Topic中的一条特定的消息，只会被订阅此Topic的每个group中的其中一个consumer消费，**此消息不会发送给一个group的多个consumer**；

那么一个group中所有的consumer将会交错的消费整个Topic，每个group中consumer消息消费互相独立，我们可以认为一个group是一个"订阅"者。

* 在kafka中,**一个partition中的消息只会被group中的一个consumer消费(同一时刻)**；

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可以同时消费多个partitions中的消息。

* kafka的设计原理决定,对于一个topic，同一个group中不能有**多于**partitions个数的consumer同时消费，否则将意味着某些consumer将无法得到消息。

**kafka只能保证一个partition中的消息被某个consumer消费时是顺序的；事实上，从Topic角度来说,当有多个partitions时,消息仍不是全局有序的。**

## 3、Kafka消息的分发

**Producer客户端负责消息的分发**

* + kafka集群中的任何一个broker都可以向producer提供metadata信息,这些metadata中包含"集群中存活的servers列表"/"partitions leader列表"等信息；
  + 当producer获取到metadata信息之后, producer将会和Topic下所有partition leader保持socket连接；
  + 消息由producer直接通过socket发送到broker，中间不会经过任何"路由层"，事实上，消息被路由到哪个partition上由producer客户端决定；

比如可以采用"random""key-hash""轮询"等,**如果一个topic中有多个partitions,那么在producer端实现"消息均衡分发"是必要的。**

* 在producer端的配置文件中,开发者可以**指定partition**路由的方式。

**Producer消息发送的应答机制**

设置发送数据是否需要服务端的反馈,有三个值0,1,-1

0: producer不会等待broker发送ack

1: 当leader接收到消息之后发送ack

-1: 当所有的follower都同步消息成功后发送ack

request.required.acks=0

## ****4、********Consumer的负载均衡****

当一个group中,有consumer加入或者离开时,会触发partitions均衡.均衡的最终目的,是提升topic的并发消费能力，步骤如下：

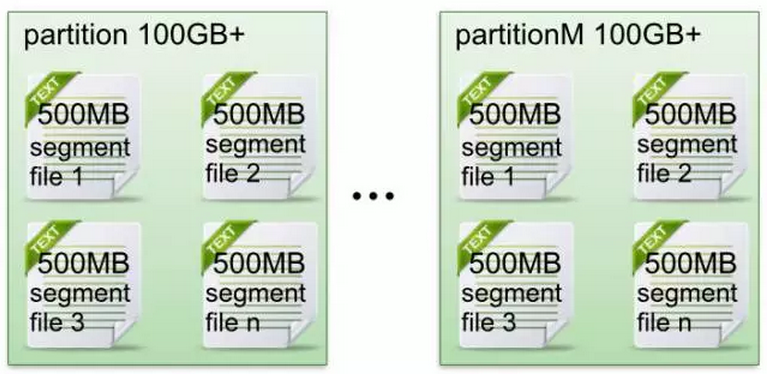
1. 假如topic1,具有如下partitions: P0,P1,P2,P3
2. 加入group中,有如下consumer: C1,C2
3. 首先根据partition索引号对partitions排序: P0,P1,P2,P3
4. 根据consumer.id排序: C0,C1
5. 计算倍数: M = [P0,P1,P2,P3].size / [C0,C1].size,本例值M=2(向上取整)
6. 然后依次分配partitions: C0 = [P0,P1],C1=[P2,P3],即Ci = [P(i \* M),P((i + 1) \* M -1)]  
   **图是错的：p2 和p3反了**



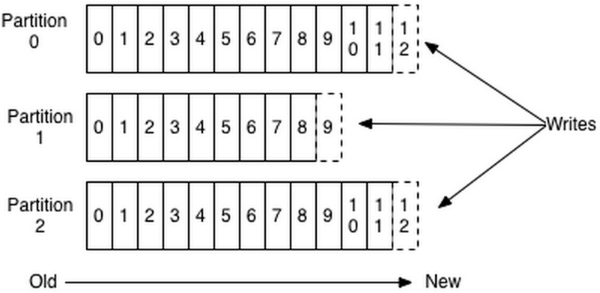
## 5、kafka文件存储机制

### 5.1、Kafka文件存储基本结构

* 在Kafka文件存储中，同一个topic下有多个不同partition，每个partition为一个目录，partiton命名规则为topic名称+有序序号，第一个partiton序号从0开始，序号最大值为partitions数量减1。
* 每个partion(目录)相当于一个巨型文件被平均分配到多个大小相等**segment**(段)数据文件中。**但每个段segment file消息数量不一定相等**，这种特性方便old segment file快速被删除。默认保留7天的数据。



* 每个partiton只需要支持顺序读写就行了，segment文件生命周期由服务端配置参数决定。（什么时候创建，什么时候删除）



数据有序的讨论？

一个partition的数据是否是有序的？ 间隔性有序，不连续

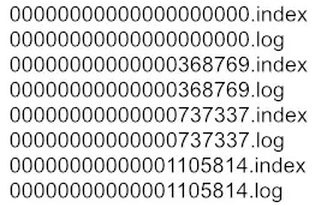
**针对一个topic里面的数据，只能做到partition内部有序，不能做到全局有序。**

特别加入消费者的场景后，如何保证消费者消费的数据全局有序的？**伪命题。**

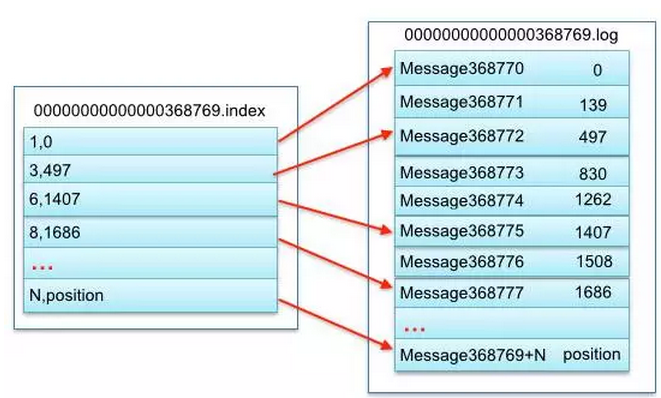
**只有一种情况下才能保证全局有序？就是只有一个partition**。

### 5.2、Kafka Partition Segment

* Segment file组成：由2大部分组成，分别为index file和data file，此2个文件一一对应，成对出现，后缀".index"和“.log”分别表示为segment索引文件、数据文件。**默认大小为1g**



* 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segment文件最后一条消息的**offset**值。数值最大为64位long大小，19位数字字符长度，没有数字用0填充。
* **索引文件存储大量元数据，数据文件存储大量消息**，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**。



**3，497：当前log文件中的第几条信息，存放在磁盘上的那个地方**

上述图中索引文件存储大量元数据，数据文件存储大量消息，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。

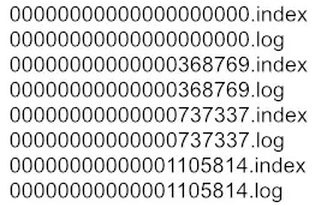
其中以索引文件中元数据3,497为例，依次在数据文件中表示第3个message(**在全局partiton表示第368772个message**)、以及该消息的物理偏移地址为497。

* segment data file由许多message组成， qq物理结构如下：

| **关键字** | **解释说明** |
| --- | --- |
| 8 byte offset | 在parition(分区)内的每条消息都有一个有序的id号，这个id号被称为偏移(offset),它可以唯一确定每条消息在parition(分区)内的位置。**即offset表示partiion的第多少message** |
| 4 byte message size | message大小 |
| 4 byte CRC32 | 用crc32校验message |
| 1 byte “magic" | 表示本次发布Kafka服务程序协议版本号 |
| 1 byte “attributes" | 表示为独立版本、或标识压缩类型、或编码类型。 |
| 4 byte key length | 表示key的长度,当key为-1时，K byte key字段不填 |
| K byte key | 可选 |
| value bytes payload | 表示实际消息数据。 |

### 5.3、Kafka 查找message

读取offset=368776的message，需要通过下面2个步骤查找。



#### 5.3.1、查找segment file

00000000000000000000.index表示最开始的文件，起始偏移量(offset)为0

00000000000000368769.index的消息量起始偏移量为368770 = 368769 + 1

00000000000000737337.index的起始偏移量为737338=737337 + 1

其他后续文件依次类推。

以起始偏移量命名并排序这些文件，只要根据offset \*\*二分查找\*\*文件列表，就可以快速定位到具体文件。当offset=368776时定位到00000000000000368769.index和对应log文件。

#### 5.3.2、通过segment file查找message

当offset=368776时，依次定位到00000000000000368769.index的元数据物理位置和00000000000000368769.log的物理偏移地址

然后再通过00000000000000368769.log顺序查找直到offset=368776为止。

## 6、Kafka自定义Partition

见代码