**Kafka的个人理解和总结**

kafka是一个提供统一的、高吞吐、低延迟的，用来处理实时数据的流式平台，它具备以下三特性：

1.流式记录的发布和订阅：类似于消息系统。

2存储：在一个分布式、容错的集群中安全持久化地存储流式数据。

3处理：编写流处理应用程序，对实时事件进行响应。

Kafka的设计原理：

kafka中是以**Topic**机制来对消息进行分类的，**同一类消息属于同一个Topic**，可以将每个Topic看成是一个消息队列。生产者将消息发送到相应的Topic，而消费者通过从Topic拉取消息来消费，**在kafka中是要求消费者主动拉取消息消费的，它并不会主动推送消息。**

**Partition分区，**每个topic可以有多个分区，这是kafka为了提高并发量而设计的一种机制：一个topic下的多个分区可以并发接收消息，同样的也能供消费者并发拉取消息，即分区之间互不干扰，这样的话，有多少个分区就可以有多大的并发量。所以，如果要更准确的打比方，**一个分区就是一个消息队列，只不过这些消息队列同属于一种消息分类。**

在kafka服务器，分区是以目录形式存在的，每个分区目录中，kafka会按配置大小或配置周期将分区拆分成多个段文件(LogSegment), 每个段由三部分组成：

1.磁盘文件：\*.log (用于存储消息本身的数据内容)

2.位移索引文件：\*.index (存储消息在文件中的位置(包括消息的逻辑offset和物理存储offset))

3.时间索引文件：\*.timeindex (存储消息创建时间和对应逻辑地址的映射关系。)

将分区拆分成多个段是为了控制存储的文件大小，如果整个分区只保存为一个文件，那随着分区里消息的增多，文件也将越来越大，最后不可控制。而如果每个消息都保存为一个文件，那文件数量又将变得巨大，同样容易失去控制。所以kafka采用段这种方式，控制了每个文件的大小，也方便控制所有文件的数量。同时，这些文件因为大小适中，**可以很方便地通过操作系统mmap机制映射到内存中，提高写入和读取效率。**这个设计的另一个好处是：当系统要清除过期数据时，可以直接将过期的段文件删除，非常简洁。

但是这里也会有一个问题：如果每个消息都要在index文件中保存位置信息，那么index文件也很容易变得很大，这样又会减弱上文所说的好处。所以在kafka中，index设计为稀疏索引来降低index的文件大小，这样，index文件存储的实际内容为：**该段消息在消息队列中的相对offset和在log文件中的物理偏移量映射的稀疏记录。**

通过系统配置来进行设置。索引记录固定为8个字节大小，分别为4个字节的相对offset(消息在partition中全局offset减去该segment的起始offset),4个字节的消息具体存储文件的物理偏移量。

kafka的查询机制：

Kafka不会在消费者拉取完消息后马上就清理消息，而是会保存段文件一段时间，直到其过期再标记为可清理，由后台程序定期进行清理。****这种机制使得消费者可以重复消费消息，满足更灵活的需求。****

kafka虽然作为消息系统，但是消费消息并不是通过推送而是通过拉取来消费的，****client需要通过offset和size参数主动去查询消息。****

kafka收到客户端请求后，对消息的寻址会经过下面几个步骤:

1.查找具体的Log Segment，kafka将段信息缓存在跳跃表中，所以这个步骤将从跳跃表中获取段信息。

2.根据offset在index文件中进行定位，找到匹配范围的偏移量position，此时得到的是一个近似起始文件偏移量。

3.从Log文件的position位置处开始往后寻找，直到找到offset处的消息。

Kafka的两大应用：

1.建立实时流数据管道，在系统或应用之间实时地传输数据。

2.构建对数据流进行转换和处理的实时流应用程序。

**Kafka与springboot的整合：**

导入依赖：

<dependency>

<groupId>org.springframework.kafka</groupId>

<artifactId>spring-kafka</artifactId>

<version>1.1.1.RELEASE</version>

</dependency>

生产者生产消息：

public class UserLogProducer {

@Autowired

private KafkaTemplate kafkaTemplate;

public void sendLog(String userid){

UserLog userLog = new UserLog();

userLog.setUsername("jhp").setUserid(userid).setState("0");

System.err.println("发送用户日志数据:"+userLog);

kafkaTemplate.send("user-log", JSON.toJSONString(userLog));

}

}

消费者消费消息：

public class UserLogConsumer {

@KafkaListener(topics = {"user-log"})

public void consumer(ConsumerRecord<?,?> consumerRecord){

//判断是否为null

Optional<?> kafkaMessage = Optional.ofNullable(consumerRecord.value());

log.info(">>>>>>>>>> record =" + kafkaMessage);

if(kafkaMessage.isPresent()){

//得到Optional实例中的值

Object message = kafkaMessage.get();

System.err.println("消费消息:"+message);

}

}