方式1（1.3版本以前）：

.数据每200ms生成一个block（存于Executor里）

如果遇到特殊情况，程序停止了，就可能出现数据重复处理或者漏处理

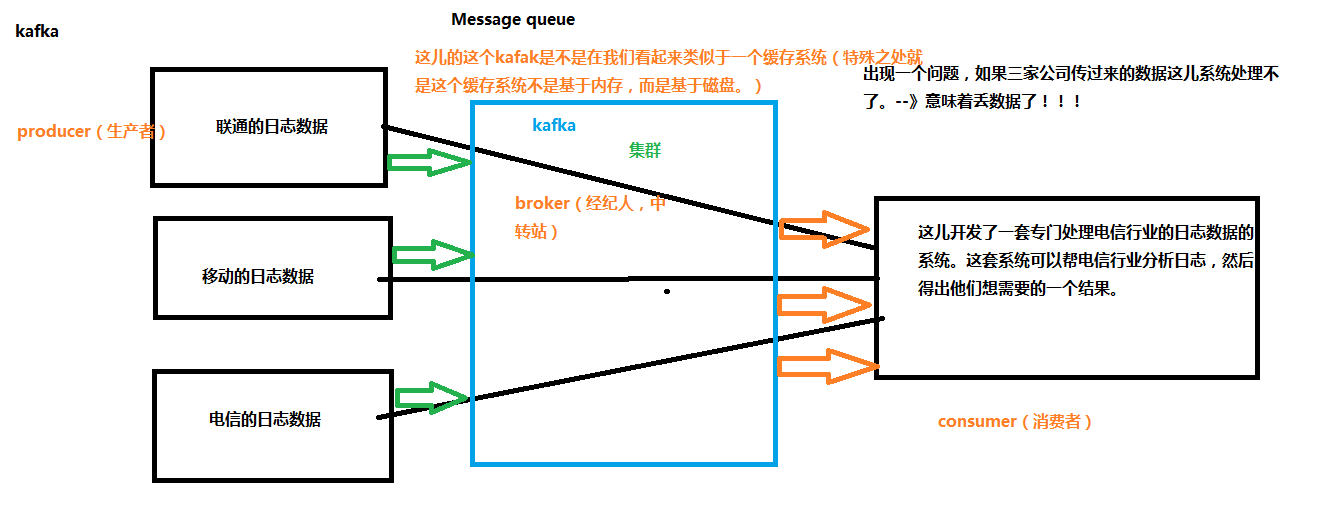
方式2（现在版本）：

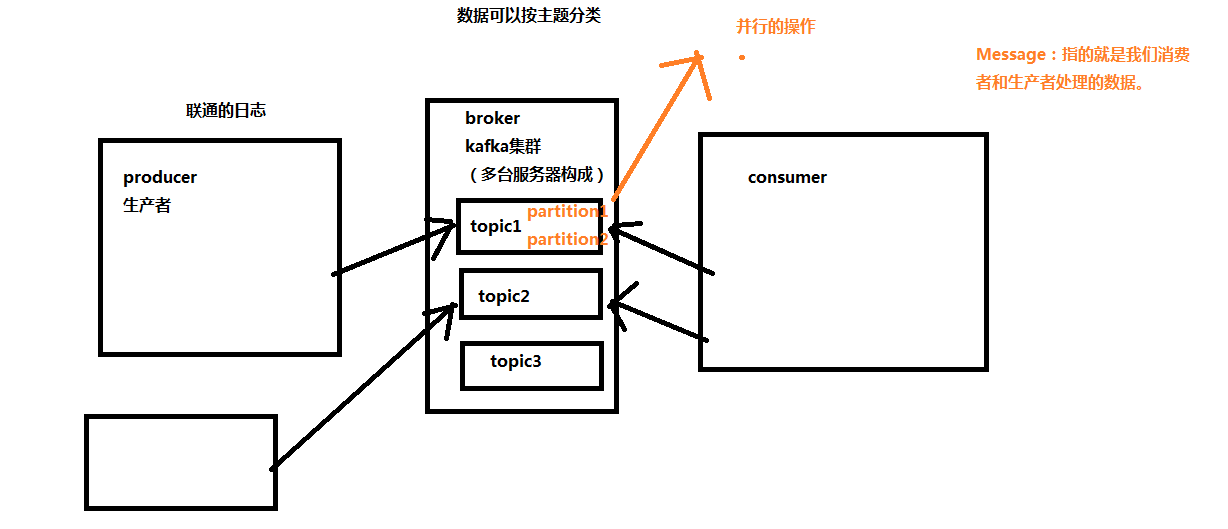
数据流进来先分配一个偏移量，处理完数据就更新偏移量，要是停电了也还是从没消费的偏移量开始处理，数据不会丢失和重复。

Kafka数据缓存是在“磁盘”上的。

《官方文档-spark和kafka集成有讲这个》

# Kafka原理介绍





Kafka分区相当于数据库里的分区，一样的。

它是一个发布订阅消息系统

### 消息系统

类似于qq：私聊和群聊

私聊：一对一。

群聊：一对多。

在类似于群聊，消息系统里面称为发布订阅（publish-subscribe） ps

在类似于私聊，消息系统里面称为点对点（point-to-point） p2p

以前大多数的消息系统是p2p的模式，这种模式的特点，就是数据被消费了

那么这个数据就被丢弃了，不能再被其它消费者消费。

Ps 的模式，指的是如果一个用户消费了数据，那么这个数据还在。另外一个用户可以重新消费。

在kafka里面数据是有保留周期的，默认就是一个星期。

在kafka里面是这样设计的：

User A（这是一个组）：a,b,c,d,e （组内用户） -（p2p模式）

User B（这是一个组）: f,g,h,i,j,k （组内用户）

Ps p2p两种模式的结合。

：组内是p2p模式，组与组之间（ps）

即：如果kafka里的数据被A组的b用户消费了，则A组内其它用户就不能消费了。

但是B组里的用户还可以消费。反之亦然。

所以在写程序时需要指定组名

http://kafka.apache.org/intro

# Kafka集群部署

搭建zookeeper集群：

**集群搭建步骤：**

tar -xzf kafka\_2.11-0.8.2.1.tgz

mv kafka\_2.11-0.8.2.1.tgz kafka

cd /root/lesson/kafka/config

修改配置文件server.properties

broker.id： 唯一，填数字

host.name：唯一，填服务器

zookeeper.connect=192.168.40.134:2181,192.168.40.132:2181,192.168.40.133:2181

**启动服务** nohup bin/kafka-server-start.sh config/server.properties &

**创建主题**

bin/kafka-topics.sh --create --zookeeper cmtest1:2181 --replication-factor 1 --partitions 1 --topic xiaolikafka

**查看主题**

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

**创建生产者**

bin/kafka-console-producer.sh --broker-list cmtest1:9092 --topic xiaolikafka

**创建消费者**

bin/kafka-console-consumer.sh --zookeeper localhost:2181 --topic xiaolikafka --from-beginning



package com.xtwy.streaming

import org.apache.spark.streaming.StreamingContext

import org.apache.spark.streaming.Seconds

import org.apache.spark.SparkConf

import org.apache.spark.streaming.kafka.KafkaUtils

import kafka.serializer.StringDecoder

import org.apache.spark.SparkContext

object KafkaOperation {

def main(args: Array[String]): Unit = {

val conf=new SparkConf().setMaster("local[2]").setAppName("KafkaOperation")

val sc=new SparkContext(conf);

val ssc=new StreamingContext(sc,Seconds(2));

ssc.checkpoint(".")

val kafkaParams=Map("metadata.broker.list" -> "hadoop1:9092");

val topics=Set("xtwy");

/\*\*

\* k:其实就是偏移量 offset

\* V:就是我们消费的数据

\* InputDStream[(K, V)]

\*

\* k:数据类型

\* v:数据类型

\*

\* k的解码器

\* v的解码器

\* [K, V, KD <: Decoder[K], VD <: Decoder[V]]

\*/

val kafkaDS= KafkaUtils.createDirectStream[String,String,StringDecoder,StringDecoder](ssc,kafkaParams,topics)

.map(\_.\_2)

val wordcountDS= kafkaDS.flatMap { line => line.split("\t") }

.map { word => (word,1) }

.reduceByKey(\_+\_)//window mapwithstate updatewithstateByKey topK

wordcountDS.print();

ssc.start();

ssc.awaitTermination();

}

}