# 网络爬虫

## 第一章 概念

### 1.1 是什么

网络爬虫 （web crawler），是按照一定的规则，自动地抓取万维网的信息的程序或脚本。

在大数据时代，信息的采集是一项重要的工作，而互联网中的数据是海量的，如果单纯靠人力进行信息采集，不仅低效繁琐，搜集的成本也会提高。如何自动高效地获取互联网中我们感兴趣的信息并为我们所用是一个重要的问题，而爬虫技术就是为了解决这些问题而生的。

网络爬虫(Web crawler) 也叫做网络机器人，可以代替人们自动地在互联网中进行数据信息的采集与整理。它是一种按照一定的规则，自动地抓取万维网信息的程序或者脚本，可以自动采集所有其能够访问到的页面内容，以获取相关数据。

从功能上来讲，爬虫一般分为数据采集，处理，储存三个部分。爬虫从一个或若干初始网页的URL开始，获得初始网页上的URL，在抓取网页的过程中，不断从当前页面.上抽取新的URL放入队列,直到满足系统的一定停止条件。

### 1.2 为什么有web crawler

我们初步认识了网络爬虫，但是为什么要学习网络爬虫呢?只有清晰地知道我们的学习目的，才能够更好地学习这一项知识。在此，总结了4种常见的学习爬虫的原因:。

1.可以实现搜索引擎。

我们学会了爬虫编写之后，就可以利用爬虫自动地采集互联网中的信息，采集回来后进行相应的存储或处理，在需要检索某些信息的时候，只需在采集回来的信息中进行检索，即实现了私人的搜索引擎。

2.大数据时代，可以让我们获取更多的数据源。。

在进行大数据分析或者进行数据挖掘的时候，需要有数据源进行分析。我们可以从某些提供数据统计的网站获得，也可以从某些文献或内部资料中获得，但是这些获得数据的方式，有时很难满足我们对数据的需求，而手动从互联网中去寻找这些数据，则耗费的精力过大。此时就可以利用爬虫技术，自动地从互联网中获取我们]感兴趣的数据内容，并将这些数据内容爬取回来，作为我们的数据源，再进行更深层次的数据分析，并获得更多有价值的信息。

3.可以更 好地进行搜索引擎优化(SEO)。。

对于很多SEO从业者来说，为了更好的完成工作，那么就必须要对搜索引擎的工作原理非常清楚，同时也需要掌握搜索引擎爬虫的工作原理。

而学习爬虫，可以更深层次地理解搜索引擎爬虫的工作原理，这样在进行搜索引擎优化时，才能知己知彼，百战不殆。

4.有利于就业。。

从就业来说，爬虫工程师方向是不错的选择之一，因为目前爬虫工程师的需求越来越大，而能够胜任这方面岗位的人员较少，所以属于一个比较紧缺的职业方向，并且随着大数据时代和人工智能的来临，爬虫技术的应用将越来越广泛，在未来会拥有很好的发展空间。

## 第二章 入门程序

网络爬虫 （web crawler），是按照一定的规则，自动地抓取万维网的信息的程序或脚本。

### 2.1 环境准备

JDK1.8

IDEA

## 第三章 HttpClient

网络爬虫就是用程序帮助我们访问网络上的资源，我们一直以来都是使用HTTP协议访问互联网的网页，网络爬虫需要编写程序，在这里使用同样的HTTP协议访问网页。

这里我们使用Java 的HTTP协议客户端HttpClient 这个技术，来实现抓取网页数据。

### 3.1 无参Get请求

往不存在的topic发数据也可以发，因为会默认创建一个partition和1个replication-factor

### 3.2 带参Get请求

index和log最重要

### 3.3 Post请求

### 3.4 带参数的Post请求

### 3.5连接池

如果每次请求都要创建HttpClient， 会有频繁创建和销毁的问题，可以使用连接池来解决这个问题。

测试以下代码，并断点查看每次获取的HttpClient都是不一-样的。。

public class HttpClientPooltest {  
  
  
 public static void main(String[] args) {  
 *//1.创建连接池管理器* PoolingHttpClientConnectionManager cm = new PoolingHttpClientConnectionManager();  
 *//最大连接数* cm.setMaxTotal(100);  
 *//每个主机的连接数（host，就是你访问的主机。目的：爬取数据不能总爬同一个host）* cm.setDefaultMaxPerRoute(10);  
  
  
 *//使用连接池管理发起请求  
 doGet*(cm);  
 }  
  
 private static void doGet(PoolingHttpClientConnectionManager cm) {  
 CloseableHttpClient httpClient = HttpClients.*custom*().setConnectionManager(cm).build();  
  
  
 HttpGet httpGet = new HttpGet("http://www.itcast.cn");  
  
 CloseableHttpResponse response = null;  
 try {  
 response = httpClient.execute(httpGet);  
  
 if (response.getStatusLine().getStatusCode() == 200) {  
 String content = EntityUtils.*toString*(response.getEntity(), "utf8");  
 System.*out*.println(content.length());  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }finally {  
 try {  
 response.close();  
 *//不能关闭httpClient，由连接池管理httpClient  
// httpClient.close();* } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
  
  
}

### 3.6 请求参数

有时候因为网络，或者目标服务器的原因，请求需要更长的时间才能完成，

我们需要自定义相关时间。

HttpGet httpGet = new HttpGet("http://www.itcast.cn");  
  
*//配置请求信息*RequestConfig config = RequestConfig.*custom*().setConnectTimeout(1000)*//创建链接的最长时间，单位毫秒* .setConnectionRequestTimeout(500)*//设置获取链接的最长时间，单位毫秒* .setSocketTimeout(10\*1000)*//设置数据传输的最长时间，单位毫秒* .build();  
  
httpGet.setConfig(config);  
  
CloseableHttpResponse response = null;

## 第四章 Jsoup

我们抓取到页面之后，还需要对页面进行解析。可以使用字符串处理工具解析页面，也可以使用正则表达式，但是这些方法都会带来很大的开发成本，所以我们需要使用一款专门解析html页面的技术。

### 4.1 介绍

jsoup是一款Java的HTML解析器，可直接解析某个URL地址、HTML文本内容。它提供了一套非常省力的API，可通过DOM, CSS 以及类似于jQuery 的操作方法来取出和操作数据。

jsoup的主要功能如下:。

1.从一个URL,文件或字符串中解析HTML;

2.使用DOM或CSS选择器来查找、取出数据: .

3.可操作HTML元索、属性、文本;。

### 4.2 解析

#### 4.2.1 解析url

#### 4.2.2 解析字符串

#### 4.2.3 解析文件

#### 4.2.4 使用dom方式

#### 4.2.5 使用选择器

#### 4.2.6 selector选择器

#### 4.2.7 selector选择器

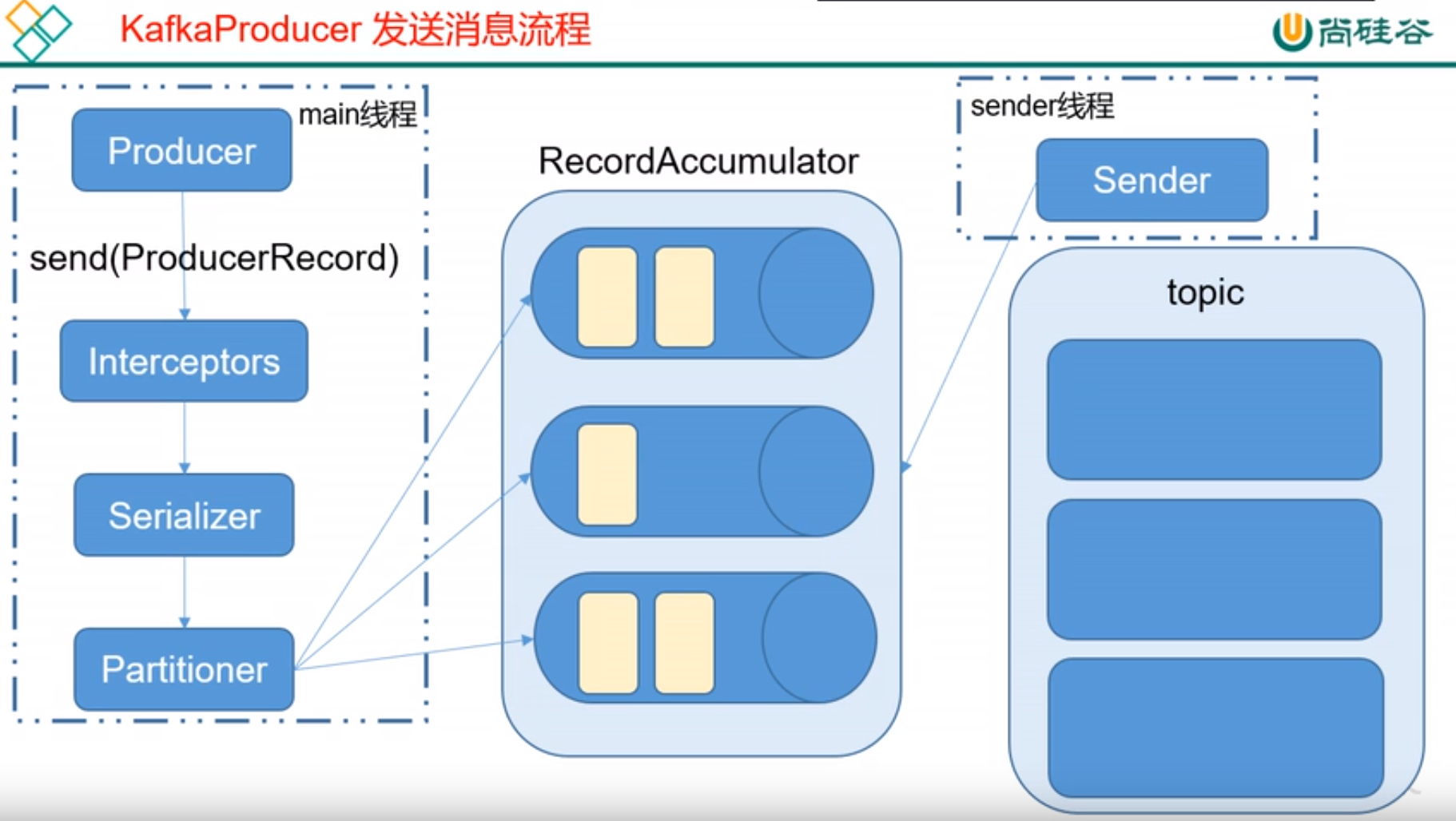
## 第五章 Kafka API

### 1 Producer API

#### 消息发送流程

Kafka的Producer发送消息采用的是异步发送的方式。两个线程：main和sender，一个线程共享变量：RecordAccumulator。

mian将消息发给RecordAccumulator，sender不断从RecordAccumulator拉取消息发送到Kafka broker。



拦截器、序列化器、分区器

batch.size：只有数据积累到batch.size之后。sender才会发送数据。

linger.ms：如果数据迟迟未到达batch.size，sender等待linger.ms之后就会发送数据

#### 1.2 异步发送API

1）导入依赖

<dependency>  
 <groupId>org.apache.kafka</groupId>  
 <artifactId>kafka-clients</artifactId>  
 <version>0.11.0.0</version>  
</dependency>

2）编写代码

需要用到的类：

**KafkaProducer：**需要创建一个生产者对象，发消息

**ProducerConfig：**获取所需的一系列配置参数

**ProducerRecord：**每条数据都要封装成年一个ProducerRecord对象

**不带回调函数的API**

import org.apache.kafka.clients.producer.KafkaProducer;  
import org.apache.kafka.clients.producer.ProducerRecord;  
  
import java.util.Properties;  
  
*/\*\*Kafka生产者  
 \* 不带回调函数的API  
 \** ***@author*** *puli2  
 \** ***@since*** *2020/1/3 16:09  
 \*/*public class MyProducer {  
  
  
 public static void main(String[] args) {  
 //1.创建Kafka producer的配置信息  
 Properties properties = new Properties();  
 //kafka集群，broker-list  
 properties.put("bootstrap.servers","172.31.63.24:9092");  
 //应答机制，at least once  
 properties.put("acks","all");  
 //重试次数  
 properties.put("retries",1);  
 //批次大小，16K  
 properties.put("batch.size",16384);  
 //逗留等待时间,单位毫秒  
 properties.put("linger.ms",1);  
 //RecordAccumulator 缓冲区大小32M  
 properties.put("buffer.memory",33554432);  
 //key\value序列化反序列化配置  
 properties.put("key.serializer","org.apache.kafka.common.serialization.StringSerializer");  
 properties.put("value.serializer","org.apache.kafka.common.serialization.StringSerializer");  
  
 KafkaProducer<String, String> producer = new KafkaProducer<String, String>(properties);  
  
 //2.发送数据,循环发送多条  
 for (int i = 0; i < 10; i++) {  
 producer.send(new ProducerRecord<String, String>("wst\_msg\_push","wst\_msg\_push--"+i));  
  
 }  
  
 //3.关闭资源  
 producer.close();  
  
 }  
}

**带回调函数的API**

//带回调函数的API  
producer.send(new ProducerRecord<String, String>("wst\_msg\_push", "wst\_msg\_push--" + i), new Callback() {  
 public void onCompletion(RecordMetadata recordMetadata, Exception e) {  
 if (e == null){  
 System.*out*.println(recordMetadata.partition()+"--"+recordMetadata.offset());  
 }else {  
 e.printStackTrace();  
 }  
 }

}

**生产者分区的API**

改成同步发送

send方法返回的是一个Future对象，根据Future对象的特点，我们也可以实现同步发送的效果，只需要调用Future对象的get方法即可。阻塞了异步的线程，达到同步

### 2 Consumer API

#### 2.1 简单消费者

import org.apache.kafka.clients.consumer.ConsumerConfig;  
import org.apache.kafka.clients.consumer.ConsumerRecord;  
import org.apache.kafka.clients.consumer.ConsumerRecords;  
import org.apache.kafka.clients.consumer.KafkaConsumer;  
  
import java.util.Arrays;  
import java.util.Properties;  
  
*/\*\*  
 \** ***@author*** *puli2  
 \** ***@since*** *2020/1/3 17:12  
 \*/*public class MyConsumer {  
  
 public static void main(String[] args) {  
 //1.创建Kafka producer的配置信息  
 Properties properties = new Properties();  
 properties.put(ConsumerConfig.*BOOTSTRAP\_SERVERS\_CONFIG*,"172.31.63.24:9092");  
 properties.put(ConsumerConfig.*ENABLE\_AUTO\_COMMIT\_CONFIG*,true);  
 properties.put(ConsumerConfig.*AUTO\_COMMIT\_INTERVAL\_MS\_CONFIG*,"1000");//自动提交offset延迟,默认开启，1000ms  
 //key\value反序列化配置  
 properties.put(ConsumerConfig.*KEY\_DESERIALIZER\_CLASS\_CONFIG*,"org.apache.kafka.common.serialization.StringDeserializer");  
 properties.put(ConsumerConfig.*VALUE\_DESERIALIZER\_CLASS\_CONFIG*,"org.apache.kafka.common.serialization.StringDeserializer");  
 //消费者组  
 properties.put(ConsumerConfig.*GROUP\_ID\_CONFIG*,"wst\_rest");  
  
 //创建消费者  
 KafkaConsumer<String, String> consumer = new KafkaConsumer<String, String>(properties);  
 //订阅主题  
 consumer.subscribe(Arrays.*asList*("wst\_msg\_push"));  
 //拉取数据  
 while (true){  
 ConsumerRecords<String, String> consumerRecords = consumer.poll(100);  
 //解析并打印ConsumerRecords  
 for (ConsumerRecord<String, String> consumerRecord : consumerRecords) {  
 System.*out*.println(consumerRecord.key()+"--"+consumerRecord.value());  
 }  
  
 }  
 }  
}

#### 2.2 重置消费者offset

一般都选

//重置消费者的offset  
properties.put(ConsumerConfig.*AUTO\_OFFSET\_RESET\_CONFIG*,"earliest");

一般都选earliest

1、只要是同一组，有一个在，再启动别的组，还是从最大的开始消费；

只要在1消费者组都没有了，重新建立的新组；2之前记录的offset因过时被删掉了

**面试题：**如何重复消费一个topic的数据

#### 2.3 手动提交offset

**后期再研究**

## 第六章 Kafka监控

## 第六章 Flume对接Kafka

## 第七章 Kafka面试题

1、Kafka中的ISP（InSyncRepli）、OSR(OutSyncRepli)、AR(AllRepli)代表什么？

2、Kafka中的HW、LEO等分别代表什么？

3、Kafka中怎么体现消息顺序性的？

4、Kafka中的分区器、序列化器、拦截器是否了解？它们之间的处理顺序是什么？

5、Kafka生产者客户端的整体结构什么样子的？使用了几个线程来处理？分别是什么？

6、“消费者组中的消费者个数如果超过topic的分区，那么就会有消费者消费不到数据”这句话是否正确？

7、消费者提交消费位移时提交的当前消费到的最新消息的offset还是offset+1？

8、有哪些情形会造成重复消费？先消费再set

9、哪些情景会造成消息漏消费？先set再消费

10、当你使用kafka-topic.sh创建（删除）了一个topic之后，Kafka背后会执行什么逻辑？

1）在zookeeper中的/brokers/topics节点下创建一个新的topic节点，如：/brokers/topics/first

2）触发Controller的监听程序

3）kafka Controller负责topic的创建工作，并更新metadata cache

11、topic的分区数可不可以增加？如果可以怎么增加？如果不可以又是为什么？

12、topic的分区数可不可以减少？如果可以怎么减少？如果不可以又是为什么？

13、kafka有内部的topic吗？如果有是什么？有什么作用？存offset

14、kafka分区分配的概念？

15、简述kafka的日志目录结构；index+log

16、如果制指定了一个offset，kafka Controller怎么查找到对应的消息？

17、聊一聊kafka Controller的作用？

18、kafka中有哪些地方需要选举？这些地方的选举策略又有哪些？controller leader

19、失效副本是指什么？有哪些应对措施？

20、kafka的哪些设计让他有如此高的性能？分布式、顺序写磁盘、0拷贝