第六届“中国软件杯”
大学生软件设计大赛

作品名称：基于 Scrapy 的电商新闻博客类网站的分布式爬虫管理系统

队 名: 雁过无声

题 号: 7

指导老师: 杨博雄

队 员: 叶彩旭 郑嘉明 潘炜桐

日 期：2017 年 6 月 30 日

目录

摘要

分布式爬虫系统作为大数据时代下一个重要的数据采集工具，无论对政府、企业都有着极其重要的作用。构建一个稳定的、高性能、高可用的分布式爬虫系统是对互联网和网站所属者之间的一场博弈，考验的双方的技术水平和能力要求。可以预见性的判断分布式爬虫系统的应用场面会越来越广泛，因为互联网信息呈爆炸性增场，可以提供大量具有时效性、参考性和科学性的数据信息。本文是基于 Scrapy 和 SpiderKeeper 框架进行的深度开发的，能对电商新闻博客类网站进行数据提取和数据结构化的分布式爬虫管理系统，能够高效、快捷、精准控制多个爬虫的运行，具备优秀的爬虫体验和表现。首先使用 scapy 框架和 scrapy-redis 组件开发分布式爬虫，redis 作为 url 队列的缓存数据库，用 Bloom-Filter 进行 url 的去重操作；采用了动态 IP 地址自动切换作为反爬虫策略，并且有研发了一个基于网页结构和文本特征正文提取算法，进行新闻博客类网站的正文提取准确率达到 95％以上，同时，能够对提取的正文信息进行数据分析，采用 TextRand 算法提取出正文的关键字和摘要信息。其次用基于 webkit 的 phantomjs 服务器作为动态网页的 JavaScript 渲染的工具，重写了底层代码实现高并发、稳定性强的 JS 渲染服务。最后是引入 Scrapyd 和 SpiderKeeper 整合分布式爬虫系统，能够监控每个爬虫的运行状态，实时控制爬虫的开始和关闭，并且监控服务器的运行情况。本文的基于 Scrapy 和 SpiderKeeper 的电商新闻博客类网站的分布式爬虫管理系统具备优秀的容错能力，在多台服务同时工作过程，能够避免由于单个爬虫出错导致系统崩溃，实现了整个分布式爬虫管理系统的高效性、稳定性和健壮性。

关键字： Scrapy 分布式 SpiderKeeper Scrapyd redis phantomjs

第一章 绪论

1.1 研究背景和意义

随着互联技术的飞速发展和网络信息数据的剧增，人们的生活和互联网的关系越来越密切相关，根据 2017 年 6 月 1 日，“互联网女皇”之称的华尔街证券分析师玛丽·米克尔（Mary Meeker）在美国 Code 大会上发布了 2017 年的互联网趋势报告。中国移动互联网用户数已破 7 亿，同比增长 12%。中国电商在全球各大市场中渗透率增长最快，占全国社会商品零售总额 15%。同时，中国在线广告收入达 400 亿美元，同比增长 30%。

获取互联网中信息最重要的来源通过搜索引擎，目前互联网上搜索引擎使用较为广泛的有：谷歌搜索，百度搜索，必应搜索等。搜索引擎的主要研究方向有:爬虫策略、数据存储、索引建立、分析算法等。网络爬虫作为搜索引擎的重要组成部分，它能对抓取的链接根据算法和策略进行排序和调度，抓取特定的内容，从而完成信息的检索，因此网络爬虫关系到搜索引擎的层与性能。

新闻博客和电商网站是网民最常使用互联网站点，网民从新闻博客能够快速获取媒体提供的信息。网购已经成为大众普遍使用的购物方式，但是海量的互联信息会隐藏一些具有欺骗性、不合法性的消息，通过网络爬虫抓取信息进行分析，可以快速发现和有效甄别不安全、不合法信息，减少谣言造成的危害，降低社会不良影响。互联网中蕴藏着大量有价值的信息，是大数据的重要来源，如何从大量网页中高效获取有价值的信息，已经成为政府和企业面临的的一个重要问题。

传统的网络爬虫只能针对个别网站进行监督，并且抓取信息的效率不高、扩展性差、维护成本高，不能运用在大型场景中。因此一个具备分布式功能的爬虫显得十分重要，能够运用多台计算机同时运行，解决由于地理位置引起的传输速率缓慢，以及面对大量数据下爬虫的性能不足等关键问题。

本文是基于 Scrapy 框架开发的分布式爬虫系统，实现高效采集新闻博客和电商网站的信息，并且进行结构化提取和保存。

1.2 爬虫简介和研究内容

网络爬虫（又称为网页蜘蛛，网络机器人），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。网络爬虫按照系统结构和实现技术，大致可以分为以下几种类型：通用网络爬虫、聚焦网络爬虫、增量式网络爬虫、深层网络爬虫。

本文研究的为聚焦网络爬虫，是指选择性地爬取根据预先定义好的主题相关页面的网络爬虫。和通用网络爬虫相比，聚焦爬虫只需要爬行与主题相关的页面，极大地节省了硬件和网络资源，保存的页面也由于数量少而更新快，还可以满足一些特定人群对特定领域信息的需求。对于现有开源的爬虫框架主要有 Nutch, Heritrix, Scrapy

(1) Nutch

Apache Nutch 是一个高度可扩展和可扩展的开源 Web 爬虫软件项目, Nutch 提供可插拔模块具有巨大优势，Nutch 提供可扩展的接口，例如 Parse，Index 和 ScoringFilter，用于自定义实现，Apache Tika 进行解析。此外，Apache Solr，弹性搜索等存在可插拔索引。Nutch 可以在单个机器上运行，也可以在 Hadoop 的分布式处理。含有搜索引擎所需的全部工具，包括全文搜索和 Web 爬虫，同时 Nutch 严格遵循 Robots 协议。

(2) Heritrix

Heritrix 是一个开源，可扩展的 web 爬虫项目。用户可以使用它来从网上抓取想要的资源。Heritrix 设计成严格按照 robots.txt 文件的排除指示和 META robots 标签。其最出色之处在于它良好的可扩展性, 方便用户实现自己的抓取逻辑。需要注意的是，Heritrix 不支持分布式，并且配置繁琐。

(3) Scrapy

Scrapy 是一个为了爬取网站数据, 提取结构性数据而编写的应用框架。可以应用在包括数据挖掘, 信息处理或存储历史数据等一系列的程序中。其最初是为了页面抓取所设计的, 也可以应用在获取 API 所返回的数据或者通用的网络爬虫。Scrapy 使用了 Twisted 异步网络库来处理网络通讯。虽然 Scrapy 框架不支持分布式爬虫，但是 Scrapy-Redis 提供一个基于 Redis 的 Scrapy 分布式组件，能够用于快速、高效开发出高可用的分布式爬虫。

文本是通过基于 Scrapy 框架整合 Scrapy-Redis 组件开发的分布式爬虫，能够对新闻博客，京东电商进行数据采集，并且设计了一套完备的应对反爬虫机制的方法和高效合理的调度策略，在性能和可用性上大大超过普通爬虫。

1.3 问题重述

Scrapy 是一个简单的 Python 爬虫框架,只需要编写几个组件就可以实现网页数据的爬取。如果需要大量爬取网页的时候,就需要使用分布式爬虫。

scrapy 原生的任务调度是基于文件系统,通过 JOBDIR 指定任务信息存储的路径。这样只能在单机执行 crawl。因此在使用 Scrapy 框架开发爬虫，需要解决面对大量网站爬取，单一爬虫性能不足导致的问题，将 Scrapy 开发一个具备分布式爬虫的功能。

- 分布式调度策略

分布式调度策略，应该将不同网站的 URL 混合后，分配到多台机器上执行。分布式调度策略的重点在 URL 的分配策略、失败处理等。分布式调度应该有多种调度策略，满足不同的场景需求。例如，有的任务必须在特定日期前执行完成，有的任务需要在另一个任务之后执行。

- URL 去重

爬虫正常运行的过程中，需要避免一个网页被多次下载，重复下载不仅会浪费 CPU 性能，还会为整个爬虫系统增加负荷。如果要管理重复性下载问题，就要考虑下载所依据的 url，只要能够管理待爬取的 URL 不重复，基本可以解决同

一个网页重复下载的问题。

- 反爬虫机制

爬虫大量爬取信息的时候，会造成服务器的负载加大，一些供应商不希望受到爬虫的干扰便会设置各种形式的反爬虫机制。并且网页数据是非常宝贵的资源，但网站运营者却要开始保护自己的数据资源，以避免被竞争对手获取到自己的数据，防止更大的商业损失。因此，爬虫需要面临的一个困难就是如何解决网站的反爬虫机制。

- 网页结构化

网页上包含大量的数据，然而却不是全部信息都是有效信息，需要从页面中准确抓取有价值的信息，并且存储进数据库是非常重要的。对于电商类网页，能对同一个网站的数据进行自动结构化，生成不同的表，例如商品表、店铺表、评价表等；对于新闻博客类网页，能进行网页正文的自动抽取，对正文进行自动摘要和关键词分析。

1.4 假设条件和说明

1.4.1 相关术语说明

本文以电商网站和新闻博客网站为抓取目标，因此抓取的内容必须能够反映电商网站和新闻博客网站的特征。

(1) 以抓取网易新闻为例，通过分析网页结构，本文确定抓取的字段如下，新闻标题，新闻来源，新闻正文，发表时间。定义的数据库字段如下:

字段名称	字段含义
News_title	新闻标题
News_source	新闻来源
News_content	新闻正文
News_date	发表时间
News_url	新闻链接

表 1:新闻提取字段

新闻标题可以作为新闻内容的关联字段，新闻来源反映新闻媒体的影响力，发表时间可以根据时间对新闻进行检索和排序，体现出新闻的时效性。

(2) 以抓取新浪博客为例，本文确定抓取的字段如下,博客作者，文章标题，发表时间，文章正文，文章来源。

字段名称	字段含义
blog_title	文章标题
blog_source	文章来源
blog_content	文章正文
blog_date	发表时间
blog_url	博客链接

表 2:博客提取字段

博客标题可以作为博客内容的关联字段，博客来源反映该博客的影响力，发表时间可以作为时间对博客文章进行时间区间检索。

(3) 京东网站作为电商网站为例，根据电商网站的特点，文本确定抓取的字段如下，商品名称，商家名称，商品价格，商品链接。

字段名称	字段含义
Product_name	商品名称
Product_store	店铺名称
Product_price	商品价格
Store_url	店铺链接
Product_comments	累计评价
Product_url	商品链接

表 3:京东提取字段

商品名称作为商品价格的关字段，商品价格和商品评价是影响消费者购买的最重要的因素，商品链接可以快速打开该商品的购买网站，查看商品详细描述。

本文描述的分布式爬虫是基于 Python3.5 版本开发的，使用的工具和框架如表格所示:

符号	说明
Python3.5	python3.5 作为开发编程语言
Scrapy	基于 python 的爬虫框架
SpiderKeeper	爬虫管理前端框架
Scrapyd	运行 scrapy 爬虫的服务程序
Redis	爬虫系统使用的缓存数据库
Scrapy-Redis	Scrapy 分布式组件
Pycharm	开发编辑器
Mysql	数据持久化的关系型数据库
phantomjs	基于 WebKit 的服务器端 JavaScript API
flask	Python 开发的 web 开发框架

表 4:相关术语解释

基于分布式系统的爬虫，本文使用了 2 台云服务器作为爬虫运行环境

机器配置 1	
操作系统	Ubuntu Server 16.04.1 LTS 64 位
CPU	2 核
内存	2GB
系统盘	20GB(本地磁盘)
公网带宽	1Mbps
IP 地址	119.29.216.186

表 5:A 服务器配置

机器配置 2	
操作系统	CentOS 7.2 64 位
CPU	1 核
内存	1GB
系统盘	20GB(本地磁盘)
公网带宽	1Mbps
IP 地址	123.207.12.105

表 6:B 服务器配置

1.4.2 Robots 网络爬虫协议

Robots 网络爬虫协议(Robots Exclusion Protocol),是指网站通过 Robots 协议告诉搜索引擎哪些页面可以抓取，哪些页面不能抓取。

Robots 协议是国际互联网界通行的道德规范，基于以下原则建立：

1、搜索技术应服务于人类，同时尊重信息提供者的意愿，并维护其隐私权；

2、网站有义务保护其使用者的个人信息和隐私不被侵犯。

当网络爬虫爬取网站时会先查看该网页根目录下是否存在 robots.txt 协议文件。但是本文的网络爬虫为了测试通用性能，故设置为忽略 Robots 协议，在此作出声明。并非是恶意爬取和下载 Robots 协议的网站，特此声明。

1.4.3 网站 Cookie

Cookie 是指一些网站为了辨别用户身份、进行 session 跟踪而储存在用户本地终端上的数据（通常经过加密）。

比如一些网站需要登录后才能访问某个页面，在登录之前，爬虫想抓取某个页面内容是不允许的。因此可以利用 Urllib2 库保存模拟登录的 Cookie，这种情况下相当于一个真实用户登录网站，并且可以正常访问网页，爬虫就可以进行网站的正常访问和下载了。

1.4.4 更新策略

电商网站的信息更新频繁，可能随时随刻都有新的信息添加，需要在特定时间内对已经抓取的网站再次抓取和更新原有信息，如果整站更新就会浪费大量资源，需要制定一套更新策略应对更新频繁的电商网站。

本文基于聚类抽样策略：网络爬虫会从电商类网站抓取标题、评论数量、时间、价格等信息。根据不同网站的特点，对网站进行区分，按照网站属性聚类，形成了不同的属性的网页类别，爬虫根据不同类别的网站抽样测试，估计网站的更新频率。爬虫会根据得到的结果，定期对网站数据进行再次抓取，更新信息。

1.5 问题分析

网络爬虫工作对带宽资源的需求大，需要解决单机爬虫在 I/O 性能上的问题，大规模爬虫爬取会涉及诸多问题：多线程并发、I/O 机制、分布式爬取、判重机制、任务调度等等，由于 Scrapy 爬虫本身不支持分布式，应对一个大型的网站资源爬取，效率和带宽会成为爬虫性能的瓶颈。为了解决单线程爬虫抓取效率问题，本文提出的一种基于 Scrapy-Redis 组件的分布式爬虫解决方案。

分布式爬虫系统设计的主要目标有以下:

(1) 高效性

互联网的网页数量庞大如海，所以爬虫的性能至关重要，这里的性能主要是指爬虫下载网页的抓取速度，常见的测试方式是以爬虫每秒能够下载的网页数量作为性能指标，单位时间能够下载的网页数量越多，则爬虫的性能越高。设计一个抓取网页的爬虫难度并不大，但是要构建一个抓取整个站点的大型网络爬虫，就需要高效的调度策略，能够对一个链接内的网站进行广度优先搜索，下载所有相关的信息。从初始的链接开始，能够识别网站中其他未被爬取的链接，加入到爬取队列，避免重复爬取相同的链接，造成性能的损失。开发分布式爬虫需要考虑到多个爬虫协同工作的问题，调度策略要根据不同爬虫的工作状态，给予不同级别的工作内容，使得多个爬虫的协同工作效率最大化。限制爬虫效率的另一个问题是内存，在带宽和内存有限的情况下，需要让爬虫实现最大效率的工作状态。

(2) 健壮性

爬虫访问不同类型的网站服务器，可能会遇到很多种非正常情况，爬取过程中无法确保每次请求都是返回正确的结果，要提高健壮性，能对错误数据、超时、程序死锁、网页 HTML 编码不规范，被抓取服务器死机，甚至是爬虫陷阱等都能进行处理，才能确保爬虫在长时间内运行。多个爬虫如果其中一个出现异常，可能会影响整个系统的正常运行，因此需要设计出具有高容错能力的爬虫。比如，爬虫对各种异常情况能够正确处理，否则可能突然停止工作。

假设爬虫程序在抓取过程中出现异常，或者爬虫所在的服务器死机，健壮的爬虫系统能够做到，再次启动爬虫时，能够恢复之前抓取的内容和数据结构，而不是每次都需要把所有工作完全从头做起，这也是爬虫健壮性的一种体现。

(3) 扩展性

当爬虫面临大量网页数量时，即使单个爬虫的性能很高，要下载所有网页的内容，需要消耗大量的时间，为了能够尽可能缩短抓取周期，爬虫系统应该有很好的可扩展性，可以便捷得通过增加抓取服务器和爬虫数量来达到此目的。现在大型网络爬虫一定是基于分布式开发的，有多台服务器专做抓取，每台服务器部署多个爬虫，每个爬虫支持多线程运行，由此来构建一个大型的网络爬虫系统。对于搜索引擎服务商来说，可能还要在全球范围、不同地域分别部署数据中心，爬虫也被分配到不同的数据中心，这样对于提高爬虫系统的整体性能是很有帮助的。

因此，扩展性好的爬虫，能够降低维护成本，面对不同的使用场景能够快速进行调整和维护。

第二章 Scrapy 框架和网络爬虫

2.1 Scrapy 框架简介

Scrapy 是一个为了爬取网站数据，提取结构性数据而编写的应用框架。 可以应用在包括数据挖掘，信息处理或存储历史数据等一系列的程序中。其最初是为了页面抓取所设计的，也可以应用在获取 API 所返回的数据或者通用的网络爬虫。

Scrapy 使用了 Twisted 异步网络库(Twisted 是用 Python 实现的基于事件驱动的网络引擎框架)来处理网络通讯，因此 Scrapy 基于并发性考虑由非阻塞(即异步)的实现。整体架构如图 1

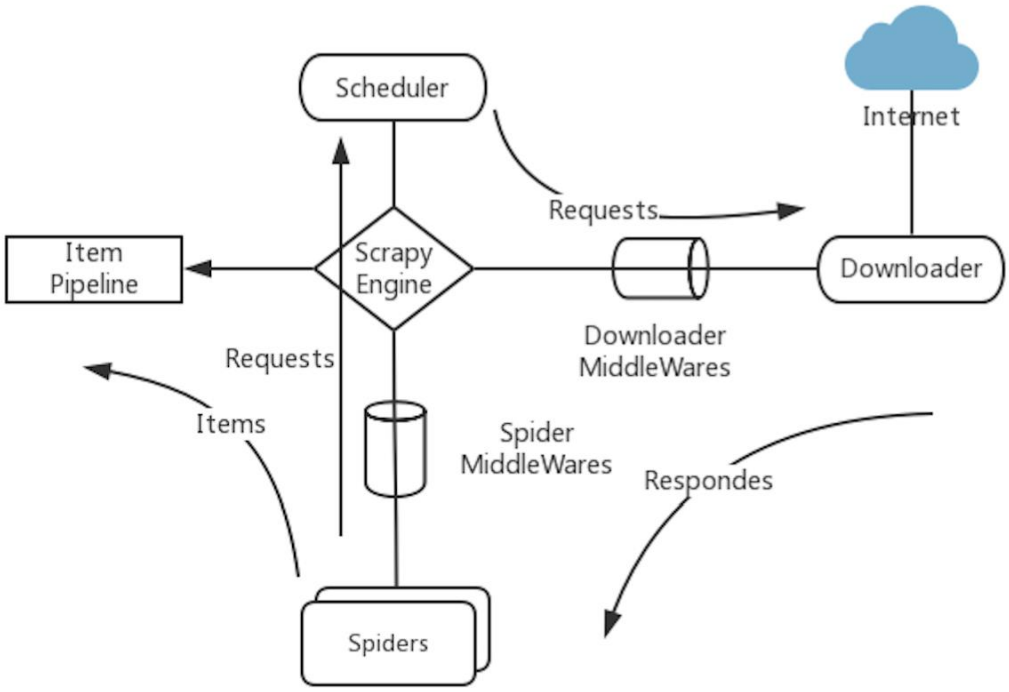


图 1:Scrapy 框架结构图

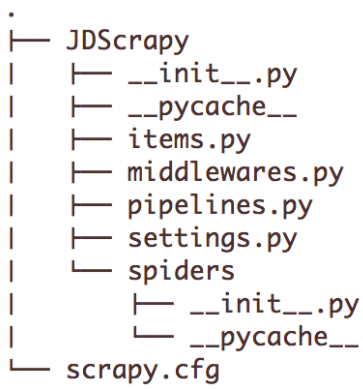


图 2: Scrapy 目录结构

2.1.1 Scrapy 主要组件

- 引擎(ScrapyEngine):

用来处理整个系统的数据流处理，触发事务(框架核心)

- 调度器(Scheduler):

用来接受引擎发过来的请求，压入队列中，并在引擎再次请求的时候返回. 可以想像成一个 URL（抓取网页的网址或者说是链接）的优先队列，由它来决定下一个要抓取的网址是什么，同时去除重复的网址

- 下载器(Downloader):

用于下载网页内容，并将网页内容返回给蜘蛛(Scrapy 下载器是建立在 twisted 这个高效的异步模型上的)

- 爬虫(Spiders):

爬虫是主要干活的，用于从特定的网页中提取自己需要的信息，即所谓的实体(Item)。用户也可以从中提取出链接,让 Scrapy 继续抓取下一个页面

- 项目管道(Pipeline):

负责处理爬虫从网页中抽取的实体，主要的功能是持久化实体、验证实体的有效性、清除不需要的信息。当页面被爬虫解析后，将被发送到项目管道，并经过几个特定的次序处理数据。

- 下载器中间件(Downloader Middlewares):

位于 Scrapy 引擎和下载器之间的框架，主要是处理 Scrapy 引擎与下载器之间的请求及响应。

- 爬虫中间件(Spider Middlewares):

介于 Scrapy 引擎和爬虫之间的框架，主要工作是处理蜘蛛的响应输入和请求输出。

- 调度中间件(Scheduler Middewares):

介于 Scrapy 引擎和调度之间的中间件，从 Scrapy 引擎发送到调度的请求和响应。

数据流(Data flow)

Scrapy 中的数据流由执行引擎控制，其过程如下:

(1) 引擎打开一个网站(open a domain)，找到处理该网站的 Spider 并向该 spider 请求第一个要爬取的 URL(s)。
(2) 引擎从 Spider 中获取到第一个要爬取的 URL 并在调度器(Scheduler)以 Request 调度。
(3) 引擎向调度器请求下一个要爬取的 URL。
(4) 调度器返回下一个要爬取的 URL 给引擎，引擎将 URL 通过下载中间件(请求(request)方向)转发给下载器(Downloader)。
(5) 一旦页面下载完毕，下载器生成一个该页面的 Response，并将其通过下载中间件(返回(response)方向)发送给引擎。
(6) 引擎从下载器中接收到 Response 并通过 Spider 中间件输入方向发送给 Spider 处理。
(7) Spider 处理 Response 并返回爬取到的 Item 及新的 Request 给引擎。
(8) 引擎将(Spider 返回的)爬取到的 Item 给 Item Pipeline，将(Spider 返回的)Request 给调度器。
(9) (从第二步)重复直到调度器中没有更多地 request，引擎关闭该网站。

2.2 Phantomjs 渲染网页

随着 JavaScript 的流行越来越多的网站选择动态加载内容，普通的爬虫发起请求，不能抓取到网页内的动态内容，因此需要结合 JavaScript 爬虫，操作 JS 获取渲染内容。目前大部分网站都是 ajax+json 获取数据的方式，一般的做法可能是抓包，接着找规律抓数据。如果碰上接口加密算法很复杂，短时间内很难破解，通过 js 抓取内容就会更加容易获取数据，因此结合 js 的爬虫就能比较直接和快捷地达到目的。

常见的结合 js 的爬虫有以下 3 种：

(1) selenium+webdriver（如 firefox，chrome 等）
这种方式要求系统有对应浏览器，并且过程中要全程开浏览器。爬虫能够抓取到和正常浏览器访问显示的数据，一般遇到特别复杂的验证码时，这个方法是有必要的，却会导致开启多个爬虫的情况下导致内存占用过多，同时影响爬虫效率和性能。

(2) scrapy-splash
splash 作为 js 渲染服务，是基于 Twisted 和 QT 开发的轻量浏览器引擎，并且提供直接的 http api。快速、轻量的特点使其容易进行分布式开发。

(3) Phantomjs
selenium +PhantomJS 是一个基于 WebKit 的服务器端 JavaScript API。它全面支持 web 而不需浏览器支持，而且加载速度快，支持各种 Web 标准： DOM 处理, CSS 选择器, JSON, Canvas, 和 SVG。PhantomJS 可以用于页面自动化，

网络监测，网页截屏 ，以及无界面测试 等。selenium 是 python 的一个第三方自动化测试库，而 phantomJS 是其子包 webdriver 下面的一个浏览器。但是 selenium 并不完美，存在许多缺点。

selenium +PhantomJS 的缺陷：

1. phantomJS 的并发问题

phantomJS 爬数据比较慢，需要提高爬虫的整体爬取效率一定需要高并发。通过多次测试下来，单个 phantomJS 程序最多支持 10 个并发，而且 phantomJS 本身在多线程方面还有很多 bug。

2. phantomJS 进程不自动关闭

每次 selenium 运行完成之后，phantomJS 进程并没有有效的正常退出。当 phantomJS 进程没有正常关闭，导致在内存中驻留的 phantomJS 进程越来越多，最终导致内存溢出。

3. 大量数据性能不稳定

phantomJS 在多进程下家在网站，会出现 JavaScript 加载失败的情况，导致无法提取出动态网页的内容，这种情况是由于 selenium 库不成熟造成的 phantomJS 性能不稳定。

既然 selenium+webdriver 存在种种缺陷，那么有没有其他方案解决呢？答案是肯定的，本文通过重写了调用 phantomJS 的 API 的程序实现的 JavaScipt 动态网页加载的，下面具体讨论 PhantomJS。

实现原理：

使用 phantomjs Webservice 作为一种 web 服务的形式（api）,将其与 python 分离开来

设计流程：

Python 通过 http 请求下发任务，Phantomjs Webservice 获取任务后去处理，处理完以后再将结果返回给 Python。任务调度、存储等复杂操作交给 Python 去做，Python 可以写成异步并发去请求 Phantomjs Webservice，需要注意的是目前一个 Phantomjs Webservice 只支持 10 个并发。但我们可以在一台服务器上多开几个 phantomjs Webservice 启用不同的端口即可，或者可以多台服务器做个集群，用 nginx 做反向代理。

代码实现:

Phantomjs Webservice

Phantomjs 是一个基于 webkit 的服务端 javascript API。phantomjs 有很多 api 接口，接口语法用的就是 js 的语法，phantom 提供了类，实例化以后可以调用对象的方法，通过回调函数可以实现自己想要的功能

Phantomjs.js

```
Var port = 8080;
try{
    //创建 page
    var webPage = require('webpage');
    var page = webPage.create();
    //清除浏览器 Cookies 信息
    phantom.clearCookies();
    //接收 json 格式的数据
    var fetch = JSON.parse(request.post);
    //设置浏览器属性
    //设置 User-Agent
    page.settings.userAgent = fetch.header['User-Agent'];
```

Phantomjs.js 作用是处理 http 请求，获取 url，获取源码操作。

Python Client.py

```
def getwebbody(self,domain):
    '''
    获取网页源代码
    '''
    base_domain=base64.b64encode(domain)
    md5_domain=hashlib.md5(base_domain).hexdigest()
    payload={domain:md5_domain}
    try:

response=requests.post(self.url,data=payload,timeout=30).content
        return response

pool = Pool(processes=10)
  for domain in domain_list:    #并发下发任务
    pool.apply_async(test, args=(domain,))
    #维持执行的进程总数为 10，当一个进程执行完后添加新进程.
  pool.close()
  pool.join()
```

Phantomjs.js 作用：异步并发下发任务

通过以上的代码，能够实现重写调用 Phantomjs 的 api 接口的程序。从而更加高效、稳定通过 Phantomjs 加载动态网页。

2.3目标网页结构分析

2.3.1 电商网站

这里以京东为例，分析电商网站的特点。京东作为国内第二大的电商平台，具有完善的商品管理系统，对商品实行了分类，同时商品数量繁多信息量庞大，具备了商品的型号，价格，商家，商品描述，评论等信息。均可以作为大数据的来源，而且京东平台使用了反爬虫机制，很适合本文的分布式爬虫测试网站。
首先，进入京东的商品首页

图 3:京东首页

京东的网页结构和各大主流的电商很相似，均是包括导航栏的商品分类，动态 js 的商品广告和热门的商品展示。实际要获取的商品详细信息，需要点击进入商品详情页查看相关信息。
以小米笔记本为例: https://item.jd.com/3312381.html

图 4:小米笔记本详情页

京东的商品 url 有固定的格式 https://item.jd.com/xxx.html，xxx 即为商品 ID，在商品详情页可以获得商品名称，商品价格，商家，评论数等一系列信息。值得注意的是，有相当一部分信息是京东为了反爬虫而使用了动态加载的技术，价格和评论的数据是通过 ajax 请求得到，或者通过 JavaScript 生成的。可以打开 chrome 的调试模式查看 ajax 的请求。

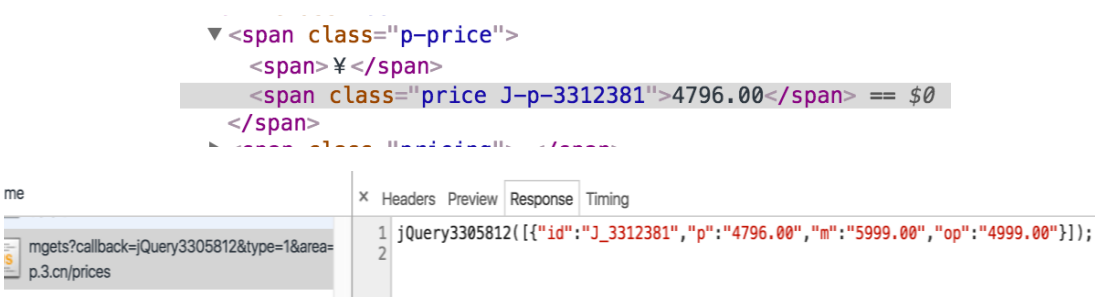


图 5 :chrome 控制台查看 js 加载的数据

通过后台的数据发现，京东的价格请求正是通过 JavaScript 生成的，如果爬虫只是靠简单的 Request 请求下载网页的内容，就无法获取商品的价格或者其他由 JavaScript 生成的数据。因此，需要解决电商中由 JavaScrtpt 生成数据的问题。

电商网站的往往包含了大量的商品图片、商品海报等图片内容，加载过程非常消耗带宽和缓存，因此对于爬虫需要抓取的信息为文字而非图片信息，可以在请求中关闭对图片信息的加载，节省爬虫的下载时间和消耗的性能，提高爬取网页的效率。

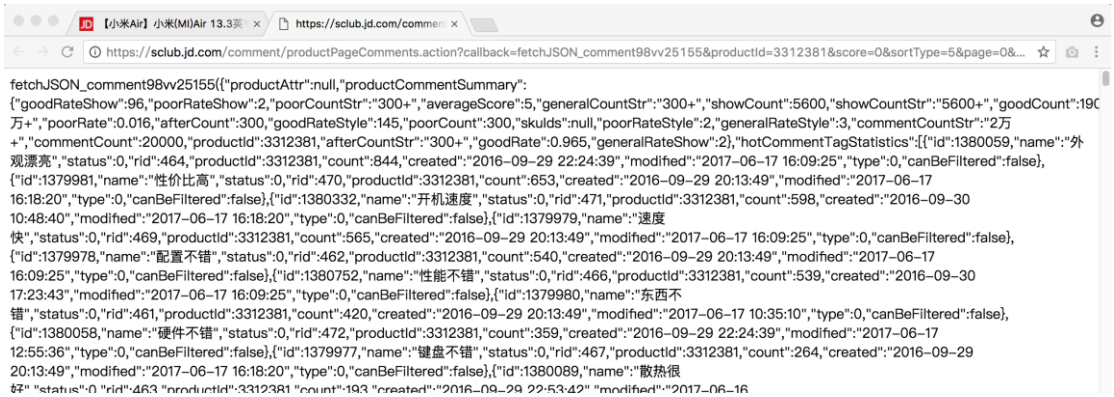


图 6:评论数据 json 格式

上图为 JavaScript 加载的评论，返回的 json 数据格式包括了买家，商品 id，评论数，评论内容，时间等信息。

对于一个电商网站，可以网页的类型归为以下几类: (1)电商首页 这里包含了整个电商的商品分类，活动详情，热门商品等信息。(2) 商品列表 这个页面就用来显示某一类别商品的搜索结果，如下图所示:从该页面可以获取这个类别商品的名称、价格、评论数、商家等信息。

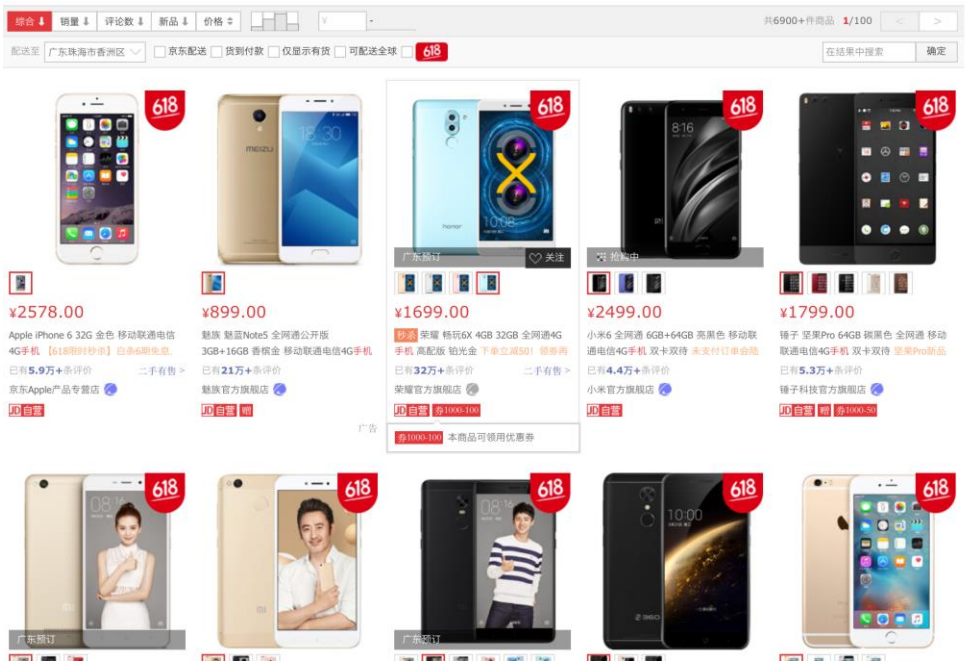


图 7:京东商品列表

(3) 商品详情页，如上图 4 所示，是爬虫需要匹配规则下载的页面，即该商品的详细介绍和说明。假设，爬虫的起始 url 为 https://www.jd.com，爬虫将会使用广度优先搜索算法，找出所有的商品详情页进行下载，完成整个电商网站的商品信息下载。目前，京东对于反爬虫的策略比较严格，例如相当一部分的内容是通过 JavaScript 生成的，如果同一个 IP 频繁访问和下载也会遭遇京东的 IP 封锁。想要获取电商大数据，务必突破以上两点。

2.3.2 电商网站提取算法

前文对京东网页结构进行分析，这里将提供对 Spider 的代码实现，完成对京东网站的数据结构化。

(1) 设置 spider 允许爬取的域，redis-key 值以及 start_urls。这里使用 BeautifulSoup 库提供的函数，对数据进行提取和分析。

(2) 下载器将下载完成的数据 Responses 返回给 Spiders。

(3) 首先使用 lxml 的解析网页，查找出所有的<a>标签，即所有的链接。由于不能保证所有的链接都是指向商品详情页面，为了提高爬虫爬爬取的策略，根据 url 的规则，只抓取商品详情页规则的 url，调整优先级下载并解析。

(4) 对于不符合规则的 url 下载并重复第三步。

JDSpider.py

```
class JdSpider(RedisSpider):
    name = "jd"
    allowed_domains = ["jd.com"]#允许爬取的范围
    redis_key = 'jd:start_urls'
    #lpush jd:start_urls https://book.jd.com/
    # 收集所有 404 的 url 以及 404 页面数
    def parse(self, response):
        #解析网页
        soup = BeautifulSoup(response.text, "lxml")
        # 获取所有的链接
        nodes = soup.find_all("a")#找到该网站的所有链接
        # 遍历链接节点
        for node in nodes: #遍历所有链接
            #链接地址
            href = str(node.get('href'))
            #写规则
            if

    len(re.findall("item.jd.com\/\d+.html$",href)) > 0:

    #寻找符合条件的 url 调整优先级下载并解析
  yield Request(url=parse.urljoin(response.url, href),callback=self.parse_detail,
priority = 10, meta={"js":1})
        elif    href.find("jd.com") >= 0:

    #不符合条件的 url 下载并重复第一步第二步
    yield Request(url=parse.urljoin(response.url, href), callback=self.parse)
```

2.4新闻博客

2.4.1 网易新闻

新闻类网站具有更新快、内容多的特点。新闻网站是大众获取信息最快捷、最有效、最权威的渠道，目前国内的大型新闻和门户网站有:新浪网、网易新闻、腾讯新闻、凤凰、搜狐以及今日头条。本文选择网易新闻作为爬取研究的对象。

图 8:网易新闻首页

网易新闻的布局结构鲜明，顶部导航栏，两侧和中间都是对热点新闻的罗列。点击标题就跳转进入新闻的详情页。

图 9 网易新闻详情页

对于新闻网站而言，爬虫需要抓取的内容均是从新闻详情页获取，包括新闻标题，发布时间，来源，摘要和新闻报道内容，一些新闻网站在网页下方还会有评论或者跟帖回复。

本文对新闻的内容提取包括的字段为:标题、正文。新闻类网站通常不需要动态加载内容，但是会有反爬虫的设定，例如：禁用 cookies, ip 大量访问会遭到限制。

通用的新闻网站爬虫，需要具备良好的降噪功能。新闻类网站页面通常有较

多的噪音，不利于对正文的提取，例如网站的导航栏，分享链接，侧栏的广告和新闻链接，评论内容，底部的版权信息，这些不需要提取的内容统称为噪音内容，在提取正文的过程中，需要把它们进行预处理和过滤。本文会介绍一种“基于节点类型和网页结构的正文提取算法”，均为自主设计和研发。能够应对常见的新闻和博客类型的正文提取，作为通用的新闻博客类通用爬虫规则。

对于博客类网站，会有一些和新闻类网站相似的网页结构，博客主要作为个人或普通媒体使用，网页结构比较同一，下面介绍的是新浪的博客类型

博客地址　http://blog.sina.com.cn/wugongdalin

图 10:新浪博客文章页

图 11: 新浪博客文章详情页

新浪博客的文章列表展示该博客的所有文章，点击链接跳转至文章内容，假设起始的 url 为博客文章列表，爬虫需要分析出所有博客文章的链接，放入待爬取队列，再从队列中取得博客文章的 url 进行下载。本文规定爬取的字段如下:标题、正文。这类规则可以作为通用的博客网站爬虫规则，博客的网站结构包括:顶部导航栏用以区分不同类型博文，左侧博主信息，可以获取博主的昵称；中间为博客文章正文，包含标题、发表时间、正文内容。底部为博客的评论区和网站版权信息。

爬虫抓取过程同样会遇到噪音信息的干扰，导致爬虫提取规则不兼容，很难提取到关键信息。因此博客和新闻类网站的提取规则均是采用了同一套方法，能够适用于主流的新闻和博客类网站的信息提取。

综上所述，新闻和博客网站采用以下的爬虫策略

1. 手动设置下载地址，通常为网站首页
2. 爬虫开始运行并从初始地址抓取第一批网站链接
3. 爬虫根据 xpath 选择器识别新链接中的新闻和博客内容地址，识别成功的新链接加入到管理 url 的队列，等待爬虫抓取，不符合要求的链接会被忽略。
4. 爬虫从待爬取队列中依次取出 url 下载并根据规则提取数据。
5. 直到队列为空，爬虫停止运行。

2.5正文提取算法

2.5.1 新闻博客提取算法

这部分主要介绍关于新闻和博客类的正文提取算法。新闻和博客类网站的重要内容为文字信息，要从中提取的主要内容为标题和正文内容。新闻网站包括新闻标题和新闻内容，博客网站包括文章标题和文章正文。本文提出一种基于网页结构和文本特征的正文提取算法进行标题和正文提取。

网页结构和主体分析

网页由 html 代码编程而成,网页的展示内容和 html 代码的位置是有联系的，都遵循至上而下的特点，通常标题会出现在 html 代码中的头部，正文内容通常出现在 html 代码的中间位置。此外，网页的布局也是遵循一定规则的，大致可以分为三个部分：顶部区域、主体区域、底部区域。顶部区域包含网站的标题，导航栏等信息；主体区域包含正文内容，同时也包含很多不相关的内容，例如作者信息，侧栏广告，评论回复等。底部区域一般用于放置版权信息等对网站不太必要的信息。

示例网站:

http://news.sina.com.cn/c/nd/2017-06-29/doc-ifyhrxtp6286676.shtml

```
<!DOCTYPE html>
<!-- [ published at 2017-06-29 00:39:44 ] -->
<!-- LLTJ_MT:name ="新浪综合" -->

<html>
<head>
  <meta charset="utf-8"/>
<meta name="sudameta" content="urlpath:c/; allCIDs:56044,257,51895,200856,51922,56261,258,38790">
<title>当着近万人的面公布手机号 这位厅官咋回事？|手机号|校长|厅官_新浪新闻</title>
```

图 11. title 在 html 的位置

```
    <div class="page-content clearfix" id="articleContent">
      <div class="left">
        <!-- 窄通 begin -->
        <ins class="sinaads" data-ad-pdps="PDPS000000056819" style="margin-top:10px;"></ins>
        <script>(sinaads = window.sinaads || []).push({});</script>
        <!-- 窄通 end -->

        <div class="article article_16" id="artibody">




    <p>　　原标题：当着近万人的面公布手机号，这位厅官咋回事？</p>
<p>　　来源：观海解局</p>
<p>　　（法制晚报记者 李洪鹏 编辑 岳三猛）网红校长“强哥”又火了：在贵州大学2017届学生毕业典礼暨学位授予仪式上，
<p>　　法晚·观海解局（微信ID：guanhaijieju）记者注意到，当时有8939名毕业生参加，且他即将赴浙江大学任职。</p
<p>　　郑强的言论受到不少网友推崇，但同时也伴随着巨大的争议。他被称为“愤青教授”、“最受大学生喜爱的校长”。喜欢他
<div class="img_wrapper"><img src="http://n.sinaimg.cn/news/transform/20170629/DnkZ-fyhneam572
<p>　　<strong>“有什么困难，我一定全力帮”</strong></p>
```

图 12:正文在 html 位置

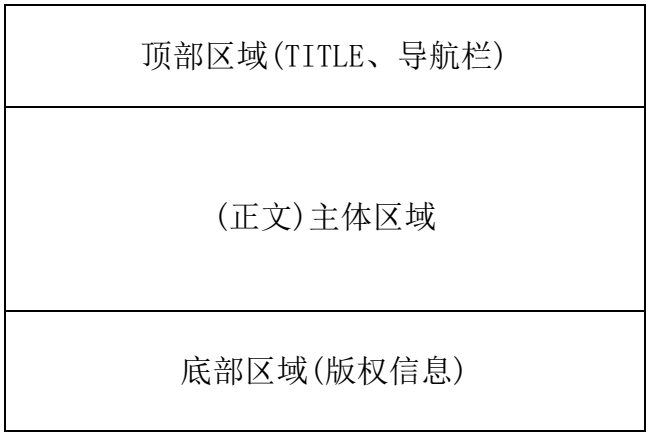


图 13：网站结构

从图 1，2 的 html 网站源代码中可以分析出 html 中标题和正文的大致位置。接着就可以进行提取算法的设计。

整体的思路大致可以分为以下几点：

1　找出网站的<title>标签，提取标签中的文字内容，确定为标题
2　根据行数和字数关系，找出两次“拐点”
3　确定正文的开始位置和结束位置
4　过滤噪音文字，提取正文

实施阶段：

根据对网页结构和文本特征的分析，可以由“文本行数（x 轴）－文本字数（y 轴）”的二维坐标轴确定正文开始和正文结束的位置。确定坐标轴的两个拐点位置，进行去头和去尾的操作。

1.网页预处理

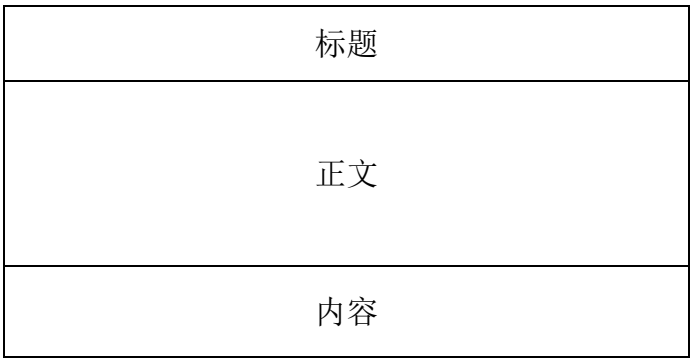


图 14：正文内容结构

通过对新闻和博客网页源代码研究和分析总结出以下噪音分类：

- CSS、JavaScript 代码块：网页的源码除了 html 标记语言，还有用语修饰网页内容的 CSS 和 JavaScript 代码。

- 注释类：<!-- … -->网站开发人员在开发过程中为了给后续维护以及便于阅读代码，而编写的注释内容。

预处理降噪过程：

1） 首先过滤掉非正文块的修饰标签，根据统计和分析得出以下的结论，CSS（层叠样式表）用于定义样式结构如字体、颜色、位置等丰富网页样式;JavaScript(动态脚本语言）常用来为网页添加各式各样的动态功能,为用户提供更流畅美观的浏览效果。通常 JavaScript 脚本是通过嵌入在 HTML 中来实现自身的功能的。本文利用 python 库中的 BeautifulSoup4 模块方法将其剔除，同时整理出以下可以忽略的节点类型

类型	标签类别
噪音节点	<script> <frame> <style> <button> <input> <select> <option>
属性	{display: none}

表 7:预处理噪音标签

2） 网页中有很大一部分重要信息存放在<a>标签中，但是，网站有一个很明显的特点，会有大量无关的标签链接指向外部的网页，例如一些广告类

的<a href=”www.hao123.com”>标签指向外部网页，因此，本文的做法是在 DOM 中寻找所有的<a>标签，根据目标抓取的网页 URL 和<a>标签的 href 属性进行匹配 Hostname(主机名)，如果不属于该域名的<a>标签，则将其过滤。

3) 编写规范的 html 代码过程中，会产生一些用于对代码进行说明，但是不会被编译和浏览器渲染的内容，即网页源代码中的注释内容，注释的内容在<!-- ….. -->标签中，这部分内容会对正文算法提取造成干扰，并且不会存放网页显示实际的数据，因此可以选择过滤<!-- .. -->标签。

4) 经过以上三个步骤的处理，已经过滤大部分的无用信息，经过降噪处理的文本，可以将其看成由两部分组成<head>和<body>部分。对于<head>部分的内容，首先应该提取出<title>部分，对于很多新闻或者博客，正文的标题就是存放在<head></head>中的<title></title>部分，需要将其提取出来。

5) 提取标题:

采用关键字匹配提取新闻和博客文章的标题<title>，由于文章的<title>格式可能为”文章标题－博客名称”的组合形式，采取字符串分割’ -- ’的方法，提取“－”前面的 title 部分。例如下图的标签遵循“标题—博客/新闻名称”格式:

6) 使用正则表达式查找所有包含<id=’footer’|id=’ft’>和<class=’footer’|class=’ft’>的行块元素。能够很大程度过滤掉博客的底部噪音。

7) 去除所有 html 标签，只保留文字信息。

至此，网页的预处理完成。

新闻
图片
博客
视频当着近万人的面公布手机号这位厅官咋回事？2017年06月29日00:14 原标题：当着近万人的面公布手机号，这位厅官咋回事？
来源：观海解局
（法制晚报记者李洪鹏编辑岳三猛）网红校长“强哥”又火了：在贵州大学2017届学生毕业典礼暨学位授予仪式上，曾任该校校长的郑强公布了自己的手机号，且表示尽力帮助大家的困难。
法晚·观海解局（微信ID：guanhaijieju）记者注意到，当时有8939名毕业生参加，且他即将赴浙江大学任职。
郑强的言论受到不少网友推崇，但同时也伴随着巨大的争议。他被称为“愤青教授”、“最受大学生喜爱的校长”。喜欢他的人，认为他特立独行，勇于改革；不喜欢他的人，认为他哗众取宠，热衷炒作。
（郑强与同学们合影）
“有什么困难，我一定全力帮”
“没想到，我跟你们同一天毕业。”6月27日，郑强在学生的欢呼声中进行了即兴演讲，“今天我就不念这个稿子了”。
他说，没有贵州，没有贵州大学，就没有“强哥”在中国的荣耀。“是贵大让我成为竖上《开讲啦》栏目的第一位中国大学校长。”在短短13分钟的演讲里，响起了12次热烈的掌声。
“今天我把手机号码告诉你们，在座的几千兄弟姐妹，强哥不一定管你们的吃住，但是你们有什么困难我能帮助的，我一定尽全力帮助。但你们别忘了一定要说一声，‘强哥’我们是同学！”
据《贵州都市报》报道，讲到深情处，57岁的郑强缓缓走到主席台前，深深地把腰弯成90度，向大家深鞠了三个躬。
在他任上，27岁能获评教授
郑强自己的“毕业”，是指他即将离任贵州大学校长，赴浙江大学任党委副书记。而4年半前来贵州大学之前，他正是浙江大学党委副书记。换句话讲，郑强像跑了个折返跑。
在贵州大学的4年多，他刮起了一场旋风：大刀阔斧推动改革，让教师别再指望课时费增收而转向科研成果，设立大学章程域予教职工代表大会、学代会、研究生代表大会更换或罢免学校各级领导干部的建议权利，引进27岁发表过世界高水平论文的博士
（郑强在毕业典礼上演讲）
法晚·观海解局（微信ID：guanhaijiej）记者梳理发现，今年57岁的郑强生于重庆。1978年，18岁的他在恢复高考的第二年，考入浙江大学化学系。
1982年大学毕业时，班里的同学有的留校，有的考上了研究生，郑强被分配到化工部某化工研究院。“单位在四川的一个山沟沟里，很偏僻。”
1985年，郑强考入成都科技大学高分子材料系硕士研究生，后留校任教。1992年被选派为中日联合培养博士生到日本京都大学留学。1995年回国后，作为引进人才回母校浙江大学任教。
2009年2月至2012年11月任浙江大学党委副书记，2012年6月起任贵州大学校长，2016年12月至今，任浙江大学党委副书记（正厅级）。
离开浙大赴贵州大学时，郑强给高分子系学生上了“最后一课”。学生们为他精心准备了一份礼物——一本电子相册，是郑强从小到大的照片，学生们花了一周时间秘密搜罗来的。
郑强眼睛瞪得老大：“哇呀！你们哪里搞来的？”他用高八度的声音说：“我经常会回来的。”头一扭又轻叹，“哎，其实是经常回不来的。”
多次呼吁加大西部教育投入
这几年，郑强一直站在风口浪尖上。他的言论受到不少网友推崇，但同时也伴随着巨大的争议。
他被称为“愤青教授”、“最受大学生喜爱的校长”。喜欢他的人，认为他特立独行，勇于改革；不喜欢他的人，认为他哗众取宠，热衷炒作。
比如，他一段关于空姐的演讲视频被指有歧视嫌疑。郑强说：“为什么中国空姐要有研究生专业？”“为什么在天上倒水的女孩要比地上倒水的长得漂亮。”郑强有点激动，离开座位，弓着腰身，学空姐推车的动作。此外，他在贵州大学任校长期间，
一个直接的结果是，离部委远的高校，自己调里肉也就少了许多——“贵州大学过去30年得到中央政府支持的总和，顶不上我原来工作浙江大学1年。”郑强说。他还打了个形象的比方：“教育部直属的重点大学都是加97号的油，我们不仅加不到93号的
责任编辑：柳龙龙文章关键词：相关阅读聚焦应用中心
新浪公益
新浪游戏
互动活动
热点推荐 · · · · ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦

新浪公司

图 15. 预处理完成效果图

预处理的结果：

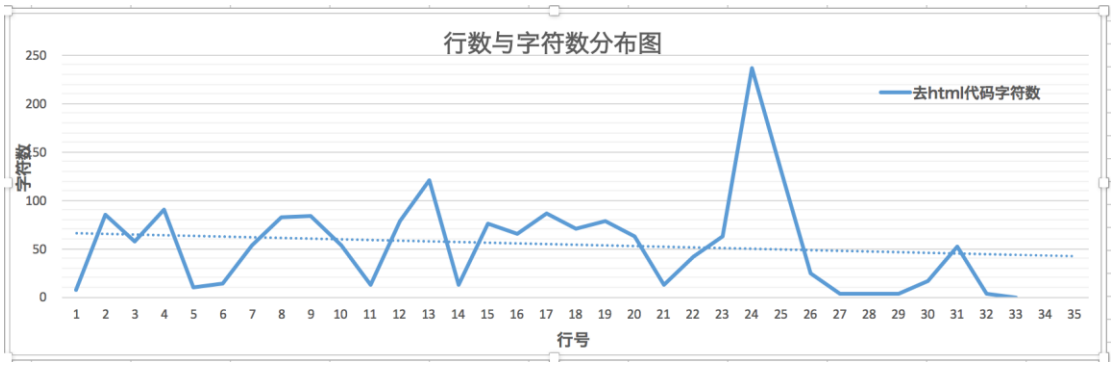


图 16 预处理后的文本与行数的关系

预处理完成之后，头部和尾部依然存在少量的噪音信息，为此本文给出以下解决方案：

本文参考了《基于行块分布函数的通用网页正文抽取》的算法，并加以改进，观察(图 1 预处理后的行数与字数的关系)效果图可知，这些噪音大部分是连续的，字数较少的，而在(X)行数-(Y)字符数的坐标轴上表示为两个拐点（参考文献《基于行块分布函数的通用网页正文抽取》）之间的 x 轴方向的距离很较长，两点之间的 Y 坐标区域较矮。所以利用以上特征进行以下过滤。

a) 找出坐标轴第一个拐点和最后一个拐点，通过以下公示计算

between = abs(currentBlock - lastBlock) / max(currentBlock, lastBlock)

between 两点之间的变化率

currentBlock　　当前行字数长度
lastBlock　　上一行字数长度

b) 如果两点之间的变化率 > 阈值（在试验中发现阈值取 0.6 时去噪效果最好），定义为拐点

c) 比较两个拐点的距离和拐点之间的每一行字符数是否满足上述说的特征，也就是说距离较长，字符数少。如果满足，可以认定它为噪音块

d) 过滤掉每个噪音块出现的文字

最终，完成对正文内容的提取。

2.6 TextRank 算法提取关键词和摘要

2.6.1 TextRank 算法介绍

TextRank 算法基于 PageRank，通过构建拓扑结构图，对词句进行排序，用于为文本生成关键字和摘要。

TextRank 算法是一种用于文本的基于图的排序算法。其基本思想来源于谷歌的 PageRank 算法，通过把文本分割成若干组成单元(单词、句子)并建立图模型，利用投票机制对文本中的重要成分进行排序，仅利用单篇文档本身的信息即可实现关键词提取、文摘。和 LDA、HMM 等模型不同，TextRank 不需要事先对多篇文档进行学习训练。

TextRank 一般模型可以表示为一个有向有权图 G =(V, E)，由点集合 V 和边集合 E 组成，E 是 V ×V 的子集。图中任两点 Vi，Vj 之间边的权重为 wji，对于一个给定的点 Vi，In(Vi) 为指向该点的点集合，Out(Vi) 为点 Vi 指向的点集合。点 Vi 的得分定义如下:

$$WS(V_i) = (1-d) + d * \sum_{V \in (V_i)} \frac{w_{ij}}{\sum_{V_k \in Out(V_j)} w_{jk}} WS(V_j)$$

其中，d 为阻尼系数，取值范围为 0 到 1，代表从图中某一特定点指向其他任意点的概率，一般取值为 0.85。使用 TextRank 算法计算图中各点的得分时，需

要给图中的点指定任意的初值，并递归计算直到收敛，即图中任意一点的误差率小于给定的极限值时就可以达到收敛，一般该极限值取 0.0001。

2.6.2 基于 TextRank 的关键词提取

关键词抽取的任务就是从一段给定的文本中自动抽取出若干有意义的词语或词组。TextRank 算法是利用局部词汇之间关系（共现窗口）对后续关键词进行排序，直接从文本本身抽取。其主要步骤如下：

(1) 把给定的文本 T 按照完整句子进行分割，即

$$T = \{ S_1, S_2, \dots, S_m \}$$

(2) 对于每个句子 $S_i \in T$，进行分词和词性标注处理，并过滤掉停用词，只保留指定词性的单词，如名词、动词、形容词，即$S_i = \{ t_{i,1}, t_{i,1}, \dots, t_{i,m} \}$，其中$t_{i,j} \in S_j$是保留后的候选关键词。

(3) 构建候选关键词图 G = (V,E)，其中 V 为节点集，由（2）生成的候选关键词组成，然后采用共现关系（co-occurrence）构造任两点之间的边，两个节点之间存在边仅当它们对应的词汇在长度为 K 的窗口中共现，K 表示窗口大小，即最多共现 K 个单词。

(4) 根据上面公式，迭代传播各节点的权重，直至收敛。

(5) 对节点权重进行倒序排序，从而得到最重要的 T 个单词，作为候选关键词。

(6) 由(5)得到最重要的 T 个单词，在原始文本中进行标记，若形成相邻词组，则组合成多词关键词。例如，文本中有句子“Matlab code for plotting ambiguity function”，如果“Matlab”和“code”均属于候选关键词，则组合成“Matlab code”加入关键词序列。

本文的关键字和摘要提取是基于 TextRank4ZH 开源项目实现的,TextRank4ZH 是针对中文文本的 TextRank 算法的 python 算法实现。

2.6.3 关键词提取

将原文本拆分为句子，在每个句子中过滤掉停用词（可选），并只保留指定词性的单词（可选）。由此可以得到句子的集合和单词的集合。

每个单词作为 pagerank 中的一个节点。设定窗口大小为 k，假设一个句子依

次由下面的单词组成：

w1, w2, w3, w4, w5, ..., wn

w1, w2, ..., wk、w2, w3, ...,wk+1、w3, w4, ...,wk+2 等都是一个窗口。在一个窗口中的任两个单词对应的节点之间存在一个无向无权的边。

基于上面构成图，可以计算出每个单词节点的重要性。最重要的若干单词可以作为关键词。

2.6.4 摘要生成

将每个句子看成图中的一个节点，若两个句子之间有相似性，认为对应的两个节点之间有一个无向有权边，权值是相似度。
通过 pagerank 算法计算得到的重要性最高的若干句子可以当作摘要。

通过测试网易新闻的正文内容
http://news.163.com/17/0623/21/CNL8DFOO000189FH.html
得出以下结果：
关键词：发展　　习近平　　人民

关键短语：农村工作　人民生活　山西建设

摘要：
6 月 21 日至 23 日，习近平在山西省委书记骆惠宁、省长楼阳生陪同下，来到吕梁、忻州、太原等地，瞻仰革命纪念馆、革命旧址，深入农村、企业，就当前经济社会发展和贯彻落实党的十八届六中全会精神进行考察调研。

第三章 Scrapy-Redis 组件和分布式

3.1 Scrapy-Redis 组件实现分布式

主从式(MasterSlave)分布式爬虫
对于主从式而言，有一台专门的 Master 服务器来维护待抓取 URL 队列，它负责每次将 URL 分发到不同的 Slave 服务器，而 Slave 服务器则负责实际的网页下载工作。Master 服务器除了维护待抓取 URL 队列以及分发 URL 之外，还要负责调解各个 Slave 服务器的负载情况。以免某些 Slave 服务器负载过大或未被充分利用。但是在这种模式下，Master 往往容易成为系统瓶颈。当系统中的爬虫

数量发生变化时，协调者需要更新地址列表里的数据，这一过程对于系统中的爬虫是透明的。但是随着爬虫网页数量的增加。控制节点会成为整个系统的瓶颈而导致整个分布式网络爬虫系统性能下降。

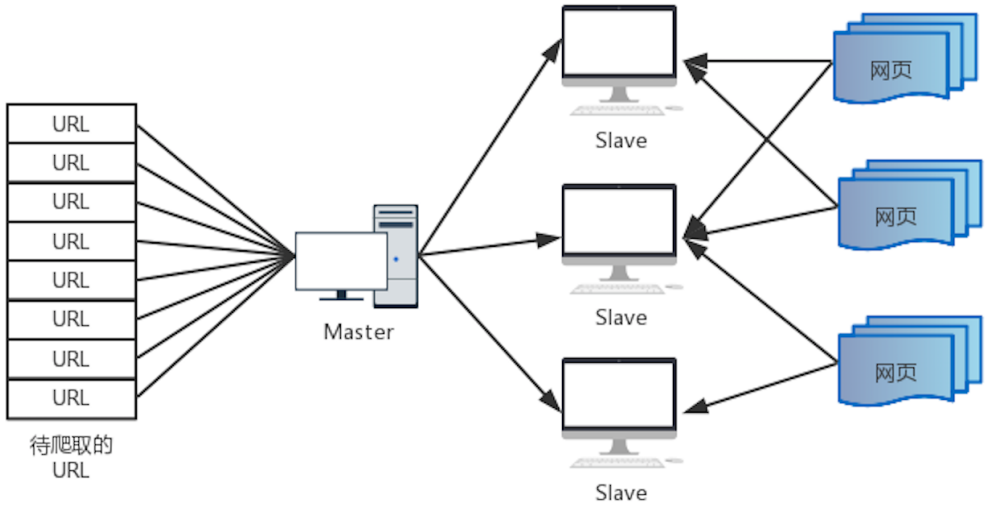


图 17.主从式分布式爬虫结构图

3.2 URL 去重策略

数据存储 NOSQL

由于分布式网络爬虫产生的数据量非常大，而且数据多为半结构化和非结构化数据。NOSQL 面对这种数据拥有良好的处理方式和存储优势。

NoSQL 数据库在以下的这几种情况下比较适用：1、数据模型比较简单；2、需要灵活性更强的 IT 系统；3、对数据库性能要求较高；4、不需要高度的数据一致性；5、对于给定 key，比较容易映射复杂值的环境。NOSQL 不遵循 ACDI 定理，它遵循 CAP 定理，即：

- 一致性(Consistency) (所有节点在同一时间具有相同的数据)
- 可用性(Availability) (保证每个请求不管成功或者失败都有响应)
- 分隔容忍(Partition tolerance) (系统中任意信息的丢失或失败不会影响系统的继续运作)

CAP 理论的核心是：一个分布式系统不可能同时很好的满足一致性，可用性和分区容错性这三个需求，最多只能同时较好的满足两个。

综合以上的概述，采用 Redis 作为 URL 队列存储的缓存数据库。

Redis 简介

Redis 是一个开源的高性能键值对数据库。它通过提供多种键值数据类型来适应不同场景下的存储需求，并借助许多高层级的接口使其可以胜任如缓存、队列系统等不同的角色。redis 先进的 key-value 存储可用于构建高性能的存储解决方案。它支持数据结构有字符串，哈希，列表，集合，带有范围查询的排序集，位图，超文本和具有半径查询的地理空间索引。

Redis 支持主从同步。数据可以从主服务器向任意数量的从服务器上同步，从服务器可以是关联其他从服务器的主服务器。这使得 Redis 可执行单层树复制。

存盘可以有意无意的对数据进行写操作。由于完全实现了发布/订阅机制，使得从数据库在任何地方同步树时，可订阅一个频道并接收主服务器完整的消息发布记录。同步对读取操作的可扩展性和数据冗余很有帮助。

Redis 优势

性能极高

- Redis 能读的速度是 110000 次/s,写的速度是 81000 次/s 。

丰富的数据类型

- Redis 支持二进制案例的 Strings, Lists, Hashes, Sets 及 Ordered Sets 数据类型操作。
- Redis 的所有操作都是原子性的，同时 Redis 还支持对几个操作全并后的原子性执行。

丰富的特性

- Redis 还支持 publish/subscribe, 通知, key 过期等等特性。

3.2 URL 去重算法

Bloom-Filter，即布隆过滤器，1970 年由 Bloom 中提出。Bloom Filter 是一种空间效率很高的随机数据结构，它利用位数组很简洁地表示一个集合，并能判断一个元素是否属于这个集合。Bloom Filter 的这种高效是有一定代价的：在判断一个元素是否属于某个集合时，有可能会把不属于这个集合的元素误认为属于这个集合（false positive）。因此，Bloom Filter 不适合那些“零错误”的应用场合。而在能容忍低错误率的应用场合下，Bloom Filter 通过极少的错误换取了存储空间的极大节省。

Bloom-Filter 的基本思想

Bloom-Filter 算法的核心思想就是利用多个不同的 Hash 函数来解决“冲突”。

计算某元素 x 是否在一个集合中，首先将所有的已知元素保存起来构成一个集合 R，然后将元素 x 跟集合 R 中的元素一一比较判断 x 是否存在于集合 R 中；通常可以采用链表等数据结构来实现。但是，随着集合 R 中元素的增加，其占用的内存将越来越大。假设有千万级别数量的不同网页需要下载，所需的内存将占用掉整个进程的内存地址空间。即使采用 MD5，UUID 将 URL 转成字符串，内存占用也是相当巨大的。

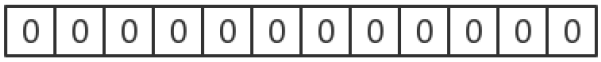
因此，可以采用 Hash Table 的数据结构，运用 Hash 函数将一个 URL 映射到二进制位数组（位图数组）中的某一位。如果该位已经被置为 1，那么表示该 URL 已经存在。

Hash 存在一个冲突（碰撞）的问题，用同一个 Hash 得到的两个 URL 的值有可能相同。为了减少冲突，可以引入多个 Hash 函数，如果通过其中的一个 Hash 值得出某元素不在集合中，那么该元素肯定不在集合中。当且仅当在所有的 Hash 函数中该元素存在集合中时，才能确定该元素存在于集合中。这就是 Bloom-Filter 的基本思想。

1）位数组：

假设 Bloom Filter 使用一个 m 比特的数组来保存信息，初始状态时，Bloom

Filter 是一个包含 m 位的位数组，每一位都置为 0，即 BF 整个数组的元素都设置为 0。

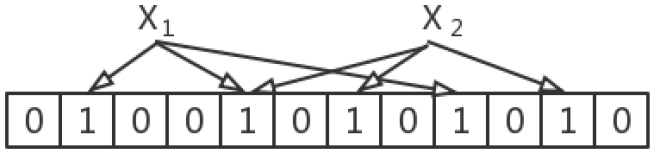


2）添加元素，k 个独立 Hash 函数

为了表达 S={x1, x2,…,xn}这样一个 n 个元素的集合，Bloom Filter 使用 k 个相互独立的哈希函数（Hash Function），它们分别将集合中的每个元素映射到{1,…,m}的范围中。

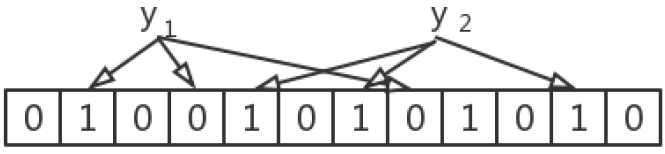
往 Bloom Filter 中增加任意一个元素 x 时候，使用 k 个哈希函数得到 k 个哈希值，然后将数组中对应的比特位设置为 1。即第 i 个哈希函数映射的位置 hashi(x)就会被置为 1（1≤i≤k）。

如果一个位置多次被置为 1，那么只有第一次会起作用，后面几次将没有任何效果。在下图中，k=3，且有两个哈希函数选中同一个位置（从左边数第五位，即第二个“1“处）。



3）判断元素是否存在集合

在判断 y 是否属于这个集合时，只需要对 y 使用 k 个哈希函数得到 k 个哈希值，如果所有 hashi(y)的位置都是 1（1≤i≤k），即 k 个位置都被设置为 1，那么就认为 y 是集合中的元素，否则就认为 y 不是集合中的元素。下图中 y1 就不是集合中的元素（因为 y1 有一处指向了“0”位）。y2 或者属于这个集合，或者刚好是一个 false positive。



虽然这个判断并不保证查找的结果是 100%正确的，但是本文的分布式爬虫正是基于 Bloom-Filter 算法实现的 URL 去重策略，尽管 Bloom-Filter 不能保证 100%正确，但是算法的高效性和准确度已经足够作为该爬虫的去重策略。

算法实现：

bloomfilter.py

Bloom Filter 参数的设定：

M　　bit 数组的宽度(bit 数)

N　　加入其中的 key 的数量

K　　使用的 hash 函数的个数

F　　False Positive 的比率

capacity　预先估计要去重的数量

error_rate错误率

conn　　redis 的连接客户端

key　　在 redis 中的键的名字前缀

```
def __init__(self, capacity=90000000, error_rate=0.00000001, conn=None,
key='BloomFilter'):
    self.m = math.ceil(capacity*math.log2(math.e)*math.log2(1/error_rate))
    #需要的总 bit 位数
    self.k = math.ceil(math.log1p(2)*self.m/capacity)
    #需要最少的 hash 次数
    self.mem = math.ceil(self.m/8/1024/1024)
    #需要的多少 M 内存
    self.blocknum = math.ceil(self.mem/512)
    #需要多少个 512M 的内存块,value 的第一个字符必须是 ascii 码，所
    有最多有 256 个内存块
```

3.3 Scrapy-redis 的基本组成及其原理

Scrapy-Redis 是基于 Scrapy 和 Redis 实现分布式的开源项目。scrapy 原生的任务调度是基于文件系统,通过 JOBDIR 指定任务信息存储的路径，这样只能在单机执行 crawl。scrapy-redis 将任务和数据信息的存取放到 redis queue ,使多台服务器可以同时执行 crawl 和 tems process,　提高了数据爬取和处理的效率。

主要功能如下:

分布式爬取

多个爬虫实例分享一个 jobs redis queue 队列, 适合大范围多域名的爬虫集群

分布式处理

爬虫抓取到的 items push 到一个 items redis queue,这就意味着可以开启多个 items processes 来处理抓取到的数据

Scrapy-Redis 主要由六个模块组成

(I) connection.py

负责根据 setting 中配置实例化 redis 连接。被 dupefilter 和 scheduler 调用，

涉及到 redis 存取的都要使用到这个模块。

(II) dupefilter.py
负责执行 requst 的去重，使用 redis 的 set 数据结构。但是注意 scheduler 并不使用其中用于在这个模块中实现的 dupefilter 键做 request 的调度，而是使用 queue.py 模块中实现的 queue。
当 request 不重复时，将其存入到 queue 中，调度时将其弹出。
(III)queue.py
其作用如 II 所述，但是这里实现了三种方式的 queue：
FIFO 的 SpiderQueue，SpiderPriorityQueue，以及 LIFI 的 SpiderStack。默认使用的是第二种。

(IV)pipelines.py
用来实现分布式处理的作用。它将 Item 存储在 redis 中以实现分布式处理。另外可以发现，同样是编写 pipelines，由于在这里需要读取配置，所以就用到了 from_crawler()函数。

(V)scheduler.py
此扩展是对 scrapy 中自带的 scheduler 的替代（在 settings 的 SCHEDULER 变量中指出），正是利用此扩展实现 crawler 的分布式调度。其利用的数据结构来自于 queue 中实现的数据结构。scrapy-redis 所实现的两种分布式：爬虫分布式以及 item 处理分布式就是由模块 scheduler 和模块 pipelines 实现。上述其它模块作为为二者辅助的功能模块。

(VI)spider.py
设计的这个 spider 从 redis 中读取要爬的 url,然后执行爬取，若爬取过程中返回更多的 url，那么继续进行直至所有的 request 完成。之后继续从 redis 中读取 url，循环这个过程。

分析：在这个 spider 中通过 connect signals.spider_idle 信号实现对 crawler 状态的监视。当 idle 时，返回新的 make_requests_from_url(url)给引擎，进而交给调度器调度。

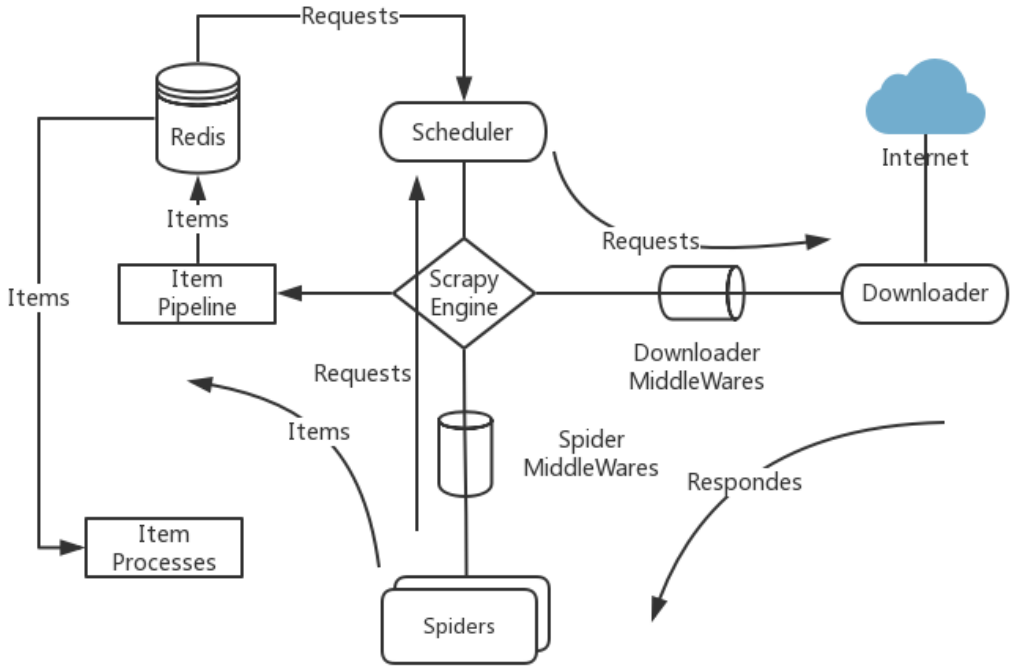


图 18:scrapy-redis 流程图

3.4 分布式调度算法

Scrapy 下载器

每一个 crawl 在正常运行中，调度器返回下一个要爬取的 URL 给引擎，引擎将 URL 通过下载中间件转发给下载器(Downloader)。下载器可以同时处理多个下载请求，在 scrapy 中主要有三个设置控制下载器的容量：

(1)CONCURRENT_REQUESTS，
(2)CONCURRENT_REQUESTS_PER_DOMAIN
(3)CONCURRENT_REQUESTS_PER_IP

第一个设置项提供了简单的下载控制，无论如何不会有超过 CONCURRENT_REQUESTS 数目的请求被并发下载。在另一方面，如果目标域名只是一个或者少数的几个，CONCURRENT_REQUESTS_PER_DOMAIN 可能就会提供对并发请求数目的更进一步的限制。假如设置了 CONCURRENT_REQUESTS_PER_IP，那么 CONCURRENT_REQUESTS_PER_DOMAIN 就被忽略掉了，此时的限制是针对每个 IP 的。在很多域名都反射一个服务器的时候，这种设置可以避免过度地访问远程服务器。

Settings.py

设置下载器的并发请求数
CONCURRENT_REQUESTS = 32
#Scrapy downloader 并发请求(concurrent requests)的最大值。默认 16 个网页同时下载

```
DOWNLOAD_DELAY = 1
#下载延迟 1S，默认是 0

CONCURRENT_REQUESTS_PER_DOMAIN = 16
#对单个网站进行并发请求的最大值为 16

CONCURRENT_REQUESTS_PER_IP = 16
# 对单个 IP 进行并发请求的最大值。如果非 0，则忽略 CONCURRENT_REQUESTS_PER_DOMAIN 设定，使用该设定。并发限制将针对 IP，而不是网站。
```

下载中间件（Downloader Middlewares）配置
下载中间件是处于引擎(crawler.engine)和下载器(crawler.engine.download())之间的一层组件，可以有多个下载中间件被加载运行。

当引擎传递请求给下载器的过程中，下载中间件可以对请求进行处理 （例如增加 http header 信息，增加 proxy 信息等）；

在下载器完成 http 请求，传递响应给引擎的过程中， 下载中间件可以对响应进行处理（例如进行 gzip 的解压等）

要激活下载器中间件组件，将其加入到 DOWNLOADER_MIDDLEWARES 设置中。该设置是一个字典(dict)，键为中间件类的路径，值为其中间件的顺序(order)。

前文提到重新设置了下载器的最大并发数量，需要在中间件组件中关闭默认下载器；反爬虫的一个策略是切换 UserAgent 伪装成不同的浏览器，在 RandomUserAgentMiddlware 中设置优先级别；JSPageMiddleware 可以设置网页的动态加载，这里设置为 0 将其关闭。

配置 DOWNLOADER_MIDDLEWARES ,数字小的先执行

```
DOWNLOADER_MIDDLEWARES = {
    # scrapy.downloadermiddlewares.useragent.UserAgentMiddleware: None,
    #关闭默认下载器
    'WuLiSpider.middlewares.RandomUserAgentMiddlware': 544,
    #随机切换 UserAgent
    'WuLiSpider.middlewares.JSPageMiddleware': 0,
    #动态加载
    'scrapy.contrib.downloadermiddleware.useragent.UserAgentMiddleware':
    None,
    }
```

爬虫在抓取网页中可能会遇到爬虫陷阱，或者一些网站无法访问而设定的重定向，将各种网络请求重新定个方向转到其它位置。为了避免这种情况的发生，需要关闭爬虫的重定向。

Settings.py

```
REDIRECT_ENABLED = False
#禁止重定向
```

3.5 分布式下高效采集

Scrapy-Redis 中的 scheduler.py 扩展是对 scrapy 中自带的 scheduler 的替代，通过扩展实现 crawler 的分布式调度。它的数据结构来自于 queue 中实现的数据结构。那么，更改 scheduler 的操作就在 settings.py 中实现，更改 SCHEDULER 的值，将原先 scrapy 的调度器变更为 scrapy-redis 的调度器。Scrapy-Redis 管理多个 spider 同时工作，爬虫引擎请求 spider 中 start_urls 中的链接并交给调度器，进而引擎向调度器请求爬取的 url 并交给下载器下载,下载后的 response 交给 spider, spider 根据定义的 rules 得到链接，继续通过引擎交给调度器。其中调度器 scheduler 中 request（url）顺序是 Redis queue 实现的，也就是将 request(url)push 到 queue 中，请求时 pop 出来，爬虫使用的默认是广度优先搜索。

Scrapy-redis 中的 queue.py 实现了三种方式的 queue：

FIFO 的 SpiderQueue，SpiderPriorityQueue，以及 LIFI 的 SpiderStack。默认使用的是第二中。在 settings.py 中设置 DUPEFILTER_CLASS 的值为 RFPDupeFilter 使用 scrapy-redis 的优先队列。
代码实现：

```
SCHEDULER = "scrapy_redis.scheduler.Scheduler"
#使用 scrapy-redis 的调度器
DUPEFILTER_CLASS = "scrapy_redis.dupefilter.RFPDupeFilter"
#使用 scrapy-redis 的优先队列
```

3.6 反爬虫策略

3.6.1 随机切换 User-Agent

从用户请求的 Headers 反爬虫是最常见的反爬虫策略。很多网站都会对 Headers 的 User-Agent 进行检测，还有一部分网站会对 Referer 进行检测（一些资源网站的防盗链就是检测 Referer）。应对这类反爬虫机制，可以直接在爬虫中添加 Headers，将浏览器的 User-Agent 复制到爬虫的 Headers 中；或者将 Referer 值修改为目标网站域名。对于检测 Headers 的反爬虫，在爬虫中修改或者添加 Headers 就能很好的绕过。
因此，在设置爬虫过程中，模拟浏览器的 Requests 请求中加入 Headers 参数，本文通过使用 fake-useragent 插件实现 User-Agent 的随机切换。
fake-useragent 能够从 useragentstring.com 抓住最新 useragent，并且在请求网站的时候随机切换，防止遭到网页反爬虫机制的限制。
代码实现：

```
class RandomUserAgentMiddlware(object):
    #随机更换 user-agent
    def __init__(self, crawler):
        super(RandomUserAgentMiddlware, self).__init__()
        self.ua = UserAgent()
        self.ua_type = crawler.settings.get("RANDOM_UA_TYPE", "random")

    @classmethod
    def from_crawler(cls, crawler):
        return cls(crawler)

    def process_request(self, request, spider):
        def get_ua():
            return getattr(self.ua, self.ua_type)
        request.headers.setdefault('User-Agent', get_ua())
```

3.6.2 禁用 cookies

Cookie 是指一些网站为了辨别用户身份、进行 session 跟踪而储存在用户本地终端上的数据（通常经过加密）。

比如一些网站需要登录后才能访问某个页面，在登录之前，爬虫想抓取某个页面内容是不允许的。因此可以利用 Urllib2 库保存模拟登录的 Cookie，这种情

况下相当于一个真实用户登录网站，并且可以正常访问网页，爬虫就可以进行网站的正常访问和下载了。

Settings.py 设置禁止 cookies

```
COOKIES_ENABLED = False
#False => 关闭 cookies
```

第四章 数据入库和爬虫管理界面

4.1 数据存储设计和数据入库

通过了上文的分布式爬虫的构建，最终要将爬虫在互联网抓取的数据储存到数据库中，进行永久存储。本文选择 Mysql 关系型数据库对抓取到的网络数据进行持久化存储。

Mysql 是最流行的开放源码 SQL 数据库管理系统，它是一种关联数据库管理系统，将数据保存在不同的表中，而不是将所有数据放在一个大的仓库内。这样就增加了速度并提高了灵活性。总体上来说，Mysql 数据库管理系统体积小、速度快、总体拥有成本低、开放源代码、性能快捷、优化 SQL 语言、容易使用、多线程和可靠性、多用户支持、可移植性和开放源代码、遵守国际标准和国际化支持、为多种编程语言提供 API。因此选择 Mysql 作为本文分布式爬虫数据存储的数据库。

4.2 设计数据库

由于本文是针对电商网站，新闻博客两类爬虫，因此设计两张表分别存储两类网站的数据

新闻类数据表：

字段名	类型	完整性约束	字段含义
id	int	PK	序号
News_title	varchar		新闻标题
News_source	varchar		新闻来源
News_content	text		新闻内容
News_date	datetime		日期
News_url	varchar	UNIQUE	新闻 URL
News_project	varchar		项目 id

表 8: 新闻类数据表

电商类数据表：

字段名	类型	完整性约束	字段含义
id	int	PK	序号
Product _name	varchar		商品名称
Product_store	varchar		店铺名称
Product _price	float		商品价格
Product_comments	varchar		累计评价
Product_url	varchar		商品链接
Store_url	varchar	UNIQUE	店铺链接
Remark	text		
date	datetime	NOT NULL	抓取时间
Project_ID	varchar		项目 id

表 9: 电商类数据表

4.3 数据入库

爬虫从网络下载数据，进而爬取从非结构性的数据源提取结构性数据，scrapy 框架通过 Items 程序控制数据的持久化操作。Item 对象是简单的容器，保存爬取得到的数据，Item 字段即 Field 对象指明每个字段的元数据。下面代码是声明京东爬取数据的 Item 结构。

items.py　部分代码

```
class JsGoodsItem(scrapy.Item):
//设置 jd_table 的 Field 字段
    id = scrapy.Field()
    name = scrapy.Field()
    url = scrapy.Field()
    price = scrapy.Field()
    date = scrapy.Field()
    remark = scrapy.Field()
    store = scrapy.Field()
    storeUrl = scrapy.Field()
    comments = scrapy.Field()
    //执行 sql 语句插入数据
    def get_insert_sql(self):
        insert_sql = """
            insert into jd_goods(id, name, url, price, remark, date,store,storeUrl,comments) VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s)
            ON DUPLICATE KEY UPDATE price=VALUES(price), date=VALUES(date)
        """
```

当爬虫从网页中抽取 Item 后，交由 Item pipeline 处理，能够验证和格式化存储数据，并经过几个特定的次序处理数据。

Item pipeline 通常执行的过程有：

- 清理 HTML 数据
- 验证解析到的数据（检查 Item 是否包含必要的字段）
- 检查是否是重复数据（如果重复就删除）
- 将解析到的数据存储到数据库中

Item pipeline 包含 from_settings、process_item、handle_error、do_insert 四个函数，接来下进行每个函数详细作用的介绍。

Pipelines.py

```
//进行配置连接到 MYSQL 数据库，部分参数在 setting 中指定
def from_settings(cls, settings):
    dbparms = dict(
        host = settings["MYSQL_HOST"],
        db = settings["MYSQL_DBNAME"],
        user = settings["MYSQL_USER"],
        passwd = settings["MYSQL_PASSWORD"],
        charset='utf8',
```

```
            cursorclass=MySQLdb.cursors.DictCursor,
            use_unicode=True,
        )
        dbpool = adbapi.ConnectionPool("MySQLdb", **dbparms)

        return cls(dbpool)

//每一个 item pipelines 都会调用该方法
def process_item(self, item, spider):
        #  使用 twisted 将 mysql 插入变成异步执行
        query = self.dbpool.runInteraction(self.do_insert, item)
        #  处理异常
        query.addErrback(self.handle_error, item, spider)
//处理异步插入的异常
def handle_error(self, failure, item, spider):
        print(failure)
//执行具体的插入
def do_insert(self, cursor, item):
        #根据不同的 item  构建不同的 sql 语句并插入到 mysql 中
        insert_sql, params = item.get_insert_sql()
        cursor.execute(insert_sql, params)
```

完成 item pipelines 组件的设置后，需要激活 Item Pipeline 组件

```
setting.py
ITEM_PIPELINES = {
    'scrapy.pipeline.PricePipeline': 300,
}
```

4.4 SpiderKeeper 和 scrapyd 的爬虫管理系统

分布式爬虫往往意味着要同时操作多台机器协同工作，能够快捷稳定的利用爬虫系统开启和终止爬取任务至关重要。关系到能否高效利用爬虫系统进行实际的爬取任务，本文是基于 Scrapyd 和 SpiderKeeper 开源爬虫管理框架进行深度开发和定制的爬虫操作控制和管理系统。

scrapyd 是运行 scrapy 爬虫的服务程序，它支持以 http 命令方式发布、删除、启动、停止爬虫程序。而且 scrapyd 可以同时管理多个爬虫，每个爬虫还可以有多个版本。爬虫发布后，会在根目录生成三个文件夹：eggs、logs、dbs，分别存放 egg 包，爬虫执行日志，爬虫数据。

Scrapyd-client 是 Scrapyd 的客户端。它提供了 scrapyd-client 和 scrapyd-deploy

实用程序，它允许将项目发布到 Scrapyd 服务器。

SpiderKeeper 是一个爬虫管理的前端框架，提供了方便的爬虫操作和实时更新查看的使用界面。

内容概要

基于新闻博客和电商网站的分布式聚焦爬虫，需要频繁进行爬虫任务的新建和操作，开发出一套面向用户和管理员的爬虫管理系统，为用户提供爬虫任务和新建、停止、查看、维护等服务。

界面设计：

图 18:爬虫管理系统界面

scrapyd 的相关 API

1、获取状态

http://127.0.0.1:6800/daemonstatus.json

2、获取项目列表

http://127.0.0.1:6800/listprojects.json

3、获取项目下已发布的爬虫列表

http://127.0.0.1:6800/listspiders.json?project=myproject

4、获取项目下已发布的爬虫版本列表

http://127.0.0.1:6800/listversions.json?project=myproject

5、获取爬虫运行状态

http://127.0.0.1:6800/listjobs.json?project=myproject

6、启动服务器上某一爬虫（必须是已发布到服务器的爬虫）

http://localhost:6800/schedule.json （post 方式，data={"project":myproject,"spider":myspider}）

7、删除某一版本爬虫
http://127.0.0.1:6800/delversion.json （ post 方 式 ，data={"project":myproject,"version":myversion}）
8、删除某一工程，包括该工程下的各版本爬虫
http://127.0.0.1:6800/delproject.json（post 方式，data={"project":myproject}）

---

scrapy 部署
将项目部署到 Scrapyd 服务器通常需要两个步骤：
1.将爬虫项目进行 Eggifying，需要借助安装 setuptools 工具。
2.通过 addversion.json 端点将 egg 上传到 Scrapyd 服务器。

scrapyd-deploy 工具自动化构建 egg 的过程并将其推送到目标 Scrapyd 服务器。使用 scrapyd-deploy <target> -p <project> 命令进行爬虫的部署，

功能介绍:
1.部署
选择部署进行爬虫任务的新建，需要提供一个 EGG 文件用于部署爬虫。完成部署后可以设置爬虫的优先级和参数。
2.控制面板
能够查看待运行任务、正运行任务、已完成任务。能够控制某个爬虫任务的运行、停止、优先级，查看日志、爬虫状态信息。
3.定期任务
增加爬虫任务，可以设置爬虫的运行时间、定时启动、优先级、参数。更加方便管理待爬取的爬虫任务。
4.爬虫运行状态
监测爬虫的运行情况，查看爬虫的线程数、抓取效率、控制台信息。
5.服务器运行状态
实时更新和显示服务器的运行状态，监测服务器的负载情况。
6.redis 管理
7.数据分析
能够对提取的新闻博客文章进行数据分析，提取正文的关键字和摘要。

4.4 性能检验和爬虫监控

为了检验本文的分布式爬虫系统的性能和可靠性，选择京东作为爬取的目标网站进行测试。

4.4.1 测试环境

由于受到设备和资金的限制，选择了三台服务器作为测试机器，其中一台 Redis 服务器作为 Master，两台服务器作为 Slave 下载网页并且提取数据存入数据库。

Master 服务器运行环境

机器配置 1	
操作系统	Ubuntu Server 16.04.1 LTS 64 位
CPU	2 核
内存	2GB
系统盘	20GB(本地磁盘)
公网带宽	1Mbps
IP 地址	119.29.216.186

表 10: Master 服务器配置

Slave 服务器运行环境

机器配置 2	
操作系统	CentOS 7.2 64 位
CPU	1 核
内存	1GB
系统盘	20GB(本地磁盘)
公网带宽	1Mbps
IP 地址	123.207.12.105

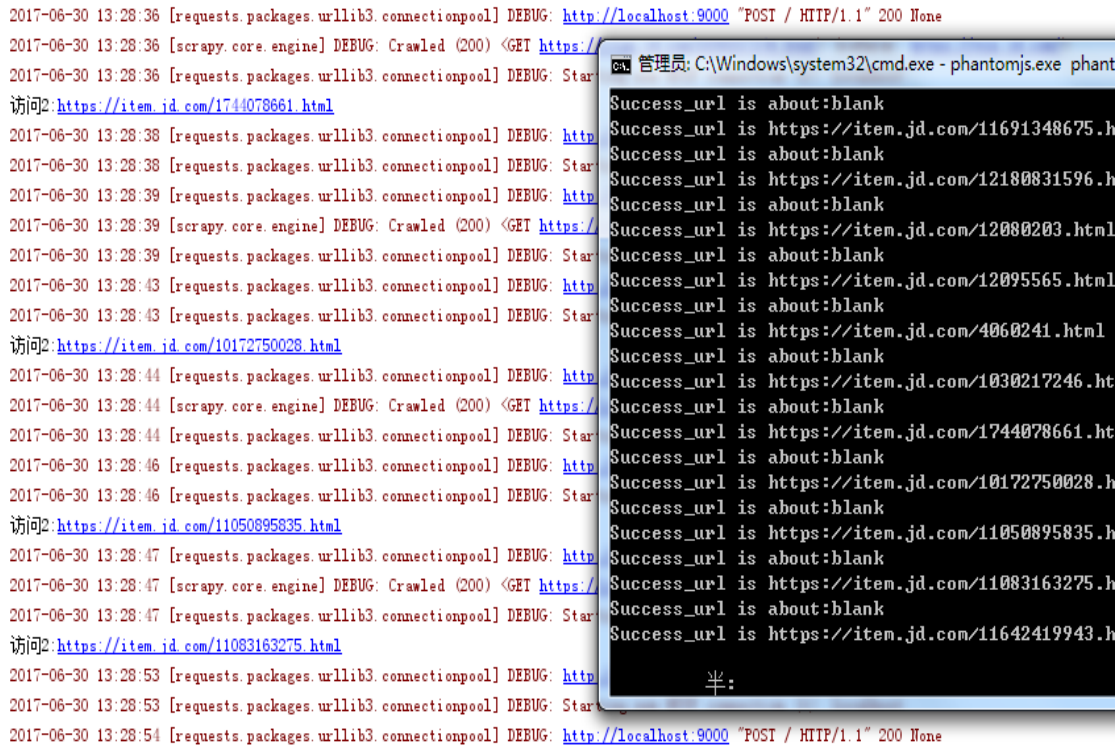
表 11: Slave 服务器配置

4.5 测试结果

4.5.1 京东电商网站测试

以京东电商作为抓取目标，在系统运行 6 小时后，分监控爬虫的运行情况、数据库的存储，最终爬取的过程和结果如下图所示：

抓取的京东商品名称和价格：



title	source	url	content
中纪委机关刊:副部官员被郭文贵指鼻子骂不作声|郭文贵|领导干部|尊严_新浪新闻	news.sina.com.cn	http://news.sina	中纪委机关刊:副部官员被
于欢缘何由无期徒刑改判为5年山东高院释疑|正当防卫|于欢|辱母杀人_新浪新闻	news.sina.com.cn	http://news.sina	于欢缘何由无期徒刑改判
北京大雨致首都机场取消航班449班|首都机场|航班|大雨_新浪新闻	news.sina.com.cn	http://news.sina	北京大雨致首都机场取消
中国影视后期产业联盟峰会成功举办48H视效预览大赛发布|华夏幸福|产业联盟|峰会_新浪新闻	news.sina.com.cn	http://news.sina	中国影视后期产业联盟峰
国内率先发布的“金融科技行业发展与法律前沿报告”|京东金融|刘鹃|中国金融_新浪新闻	news.sina.com.cn	http://news.sina	国内率先发布的“金融科
不会向任何机构提供区域和中学的排名数据|中学|排名|区域_新浪新闻	news.sina.com.cn	http://news.sina	不会向任何机构提供区域
外媒调查美学生到底多穷:9学区过半学生吃不饱|洛杉矶|学生|学区_新浪新闻	news.sina.com.cn	http://news.sina	外媒调查美学生到底多穷:
特朗普捍卫美退巴黎协定决定称对此举感到骄傲|特朗普_新浪新闻	news.sina.com.cn	http://news.sina	特朗普捍卫美退巴黎协定
菲律宾强制国民唱国歌要“热情”否则被罚款|菲律宾|唱国歌|国歌_新浪新闻	news.sina.com.cn	http://news.sina	菲律宾强制国民唱国歌要
今年国务院有七部委换帅四名“一把手”是博士|博士|何立峰|钟山_新浪新闻	news.sina.com.cn	http://news.sina	今年国务院有七部委换帅
四部门：继续适当降低企业五险一金缴费比例|实体经济|物流|缴费_新浪新闻	news.sina.com.cn	http://news.sina	四部门：继续适当降低企
土高官：将回击任何来自叙境内库尔德武装炮火|高官|土耳其|叙利亚_新浪新闻	news.sina.com.cn	http://news.sina	土高官：将回击任何来自
香港回归20年深圳海关验放供港民生品4459亿元|农产品|深圳海关|文锦渡_新浪新闻	news.sina.com.cn	http://news.sina	香港回归20年深圳海关验
堵塞交通强行闯卡:首批违法超限超载黑名单公布|超限|违法|黑名单_新浪新闻	news.sina.com.cn	http://news.sina	堵塞交通强行闯卡:首批违
宁夏自治区原书记李建华转任全国政协(图/简历)|组织部|宁夏自治区|全国政协_新浪新闻	news.sina.com.cn	http://news.sina	宁夏自治区原书记李建华

图 19:爬虫爬取的数据

最后获取的商品测试数据结果：

服务器	时间(小时)	爬取的商品数(条)
A 服务器	2	3217
A 服务器	4	6178
A 服务器	6	8780
B 服务器	2	3078
B 服务器	4	6634
B 服务器	6	8103

表 11：性能测试结果

由表（11）的数据得知，每台机器在 6 小时内爬取到的数据大约为 8000 条左右，两台服务器在六小时内爬取到的总数据为 16883 条。测试过程中限制了爬取一个网页后间隔 1s，另外可以看出整个分布式爬虫的两台机器做到负载均衡。

4.5.2 正文提取结果分析

本文算法对示例数据集给出的三个门户网站（新浪、腾讯、网易）进行测试，通过定义三个参数对算法的提取精度进行分析。评价文本提取结果的两个指标为查准率，查全率和 F1。

查准率 $P=\frac{正确分类的文本}{所有分类的文本}\times 100\%$

查全率 $R=\frac{提取正确的文本}{所有提取的文本}\times 100\%$

加权几何平均值 $F1=\frac{2PR}{P+R}$

结果分析

P 和 R 的取值都在 0 和 1 之间,数值越接近 1,查全率或查准率就越高。而 F 值是查全率和查准率的加权几何平均值,用 F 值就可表明算法的精度,F 值越接近 1 越好表示精度越高。测试 100 个网页的平均耗时为 8s。

从实验结果可以评估 Z-JM 算法在论坛正文提取的准确率和完整率都能达到 90%以上，对于不同类型的论坛(Discuz，BbsMax，PHPwind，Dvbbs，LeadBBS) 该算法能够保证较高的准确性和完整性。充分证明了本方法的兼容性、通用性。在提取过程中出现的少数错误情形，可能由于文本特征不明显，标签的使用不规范，以及过多的噪音标签，一些相关链接和正文信息同处一块或较接近，导致提取发生误差。

网页来源	查准率	查全率	F1 值
www.sina.com.cn	97.54	95.33	97.54
news.163.com	96.37	95.17	95.22
news.qq.com	95.17	94.19	94.02

表 12:提取正确率分析

第五章 总结

至此，基于 Scrapy 的电商新闻博客类网站的分布式爬虫管理系统的设计和开发工作全部完成。

5.1 工作总结

此系统有一套完备的分布式组件和抓取策略。能够爬取京东电商网站，新闻和博客类网站，并且进行结构化存储。Scrapy-Redis 组件实现的分布式，提高了原生的 Scrapy 单机爬虫的性能，URL 去重用 REDIS 作为缓存数据库和 Bloom-Filter 算法检测 URL。Scrapyd 控制爬虫开始和停止，重写了 phantomjs 服务器接口，能对 JavaScript 加载的网页进行渲染。同时提供了一套优秀的提取网页数据的算法，将网页的非结构化数据进行结构化存储到 MYSQL 数据库。

5.2 改进和展望

该分布式爬虫系统致力于抓取电商和新闻博客类网站，由于电商网站由较严格的反爬虫限制，目前只采用了几中常见反爬虫策略，假设使用动态 ip 代理进行爬虫的抓取设置，可以大幅度调整爬虫的整体爬取效率。但是动态 ip 代理需要一些资金购买，鉴于整个开发经费较少，不能够使用动态 ip 代理。

整套分布式管理爬虫系统实现京东和新闻博客类网站数据处理，虽然能够达到可用程度，完成了既定的任务和目标，但存在一些不足。首页，京东网站只需要爬取商品详情页，爬虫设定好了特定规则的 url，不能对其他类型的电商网站进行提取。对于新闻博客类网站，还不能自动识别所有的页面类型，在启动爬虫时，需要指定一个 start_url 和一个模版的 url(需要提取的网页模版 url)，才能正常提取正文内容。提高系统的实际应用能力，还有许多地方可以改善。诸如：增加多台爬虫服务器，加大系统的抓取能力；对数据进行进一步的分类处理等。

参考文献

[1] 互联网女皇 2017 年报告－新浪科技
tech.sina.com.cn/roll/2017-06-01/doc-ifyfuvpm6973304.shtml
[2] 网络爬虫-百度百科词条
[3] scrapy 教程&文档
scrapy-chs.readthedocs.io/zh_CN/
[4] Apache Nutch Wiki
https://wiki.apache.org/nutch/
[5] 设计一个高性能的分布式网络爬虫-我爱机器学习
www.52ml.net/4390.html
[6] 优秀爬虫的特征
blog.sina.com.cn/s/blog_1525836d30102wcmc.html
[7] 常见的反爬虫和应对方法
zhuanlan.zhihu.com/p/20520370
[8] Robots 网络爬虫协议 ——百度百科词条
[9] Cookie 的使用—伯乐在线
python.jobbole.com/81344/
[10] 爬虫之 scrapy-splash—scrapy+js 渲染容器
www.jianshu.com/p/2516138e9e75?open_source=weibo_search
[11]盘点 selenium phantomJS 使用的坑
www.jianshu.com/p/9d408e21dc3a
[12] Python-Scrapy 个人兴趣教程(二)：从代理 IP 开始
http://www.w2bc.com/Article/41469
[13] Phantomjs 正确打开方式
http://thief.one/2017/03/31/
[14] 谈谈主从分布式爬虫与对等分布式爬虫的优劣
www.dataguru.cn/thread-529666-1-1.html
[15] NoSQL—百度百科词条
[16] Redis 中文网
www.redis.net.cn
[17] 使用 Scrapy-redis 实现分布式爬取
www.biaodianfu.com/
[18] scrapy-redis 源码分析
http://blog.csdn.net/u012150179/article/details/38226253
[19] 海量数据处理算法—Bloom Filter
blog.csdn.net/hguisu/article/details/7866173
[20] Scrapy 性能分析
m.blog.csdn.net/Q_AN1314/article/details/51245011
[21] 爬虫框架 Scrapy 之 Downloader Middlewares
www.cnblogs.com/wzjbg/p/6507581.html

[22] scrapy-redis 实现爬虫分布式爬取分析与实现
blog.csdn.net/u012150179/article/details/38091411
[23] Textrank 算法介绍
www.cnblogs.com/clover-siyecao/p/5726480.html
[24] TextRank4ZH-Github 开源项目
github.com/letiantian/TextRank4ZH
[25] scrapyd 官方文档
scrapyd.readthedocs.io/
[26] scrapy-client 开源项目
github.com/scrapy/scrapyd-client
[27] SpiderKeeper 开源项目
github.com/DormyMo/SpiderKeeper
[28] 基于 scrapyd 爬虫发布总结
www.cnblogs.com/zhongtang/p/5634545.html

[29] 赵鹏程.分布式书籍网络爬虫系统的设计与实现[D].西南交通大学,2014.
[30] 马联帅.基于 Scrapy 的分布式网络新闻抓取系统设计与实现[D].西安电子科技大学,2015.
[31] 舒德华.基于 Scrapy 爬取电商平台数据及自动问答系统的构建[D].华中师范大学,2016.
[32] 吴霖.分布式微信公众平台爬虫系统的研究与应用[D].南华大学,2015.