南華大學
UNIVERSITY OF SOUTH CHINA

密　级：普通

文档编号：20170630-SDD　　版　次：2

分册名称：　　第 1 册/共 1 册

分布式爬虫
（20170630）
系统设计说明书

编　　写　：　刘宝玉

校　　对　：　王浩茂 杨奎

审　　核　：　刘立

批　　准　：　刘立

生效日期　：　2017 年 6 月 30 日

南华大学计算机学院制

2017 年 6 月

文件修改控制

修改记录编号	修改状态	修改页码及条款	修改人	审核人	批准人	修改日期

目录

1. 系统概述

爬虫系统，是对海量的分散的互联网数据进行采集的系统，是搜索引擎系统的基础。大数据近年来快速发展，炙手可热，不仅是数据的容量大，更是强调对全样本的数据的分析。互联网数据中包含了大量有价值信息，是大数据的重要数据来源。而互联网上的数据内容丰富，组织形式也灵活多样。传统的爬虫系统，对所有的网页采用同样的办法处理，利用深度优先或广度优先的办法获取网页链接，下载网页，对网页中的所有的文本数据建立倒排索引。这种方式没有对网页数据的信息进行组织、归类。应大数据的需求，分布式爬虫系统是解决这一问题的方案。分布式爬虫，对同一个网站的同类数据，进行结构化。同时，能利用分布式的软件设计方法，实现爬虫的高效采集。

为了方便后续的功能拓展以及使系统更加符合专业人员的需要，本小组为系统中用到的相关模型配置信息、各模型评价标准等信息提供了完整的后台配置界面，管理员可随时对相关标准和指标做出调整，满足进一步的需要。

2. 整体架构

该系统采用 B/S 结构，服务器端位于 Linux 平台，使用 Python　Django 技术。客户端使用 JavaScript，Html 以及 CSS 等技术，用户可以通过浏览器对系统进行访问。

该系统整体架构如图

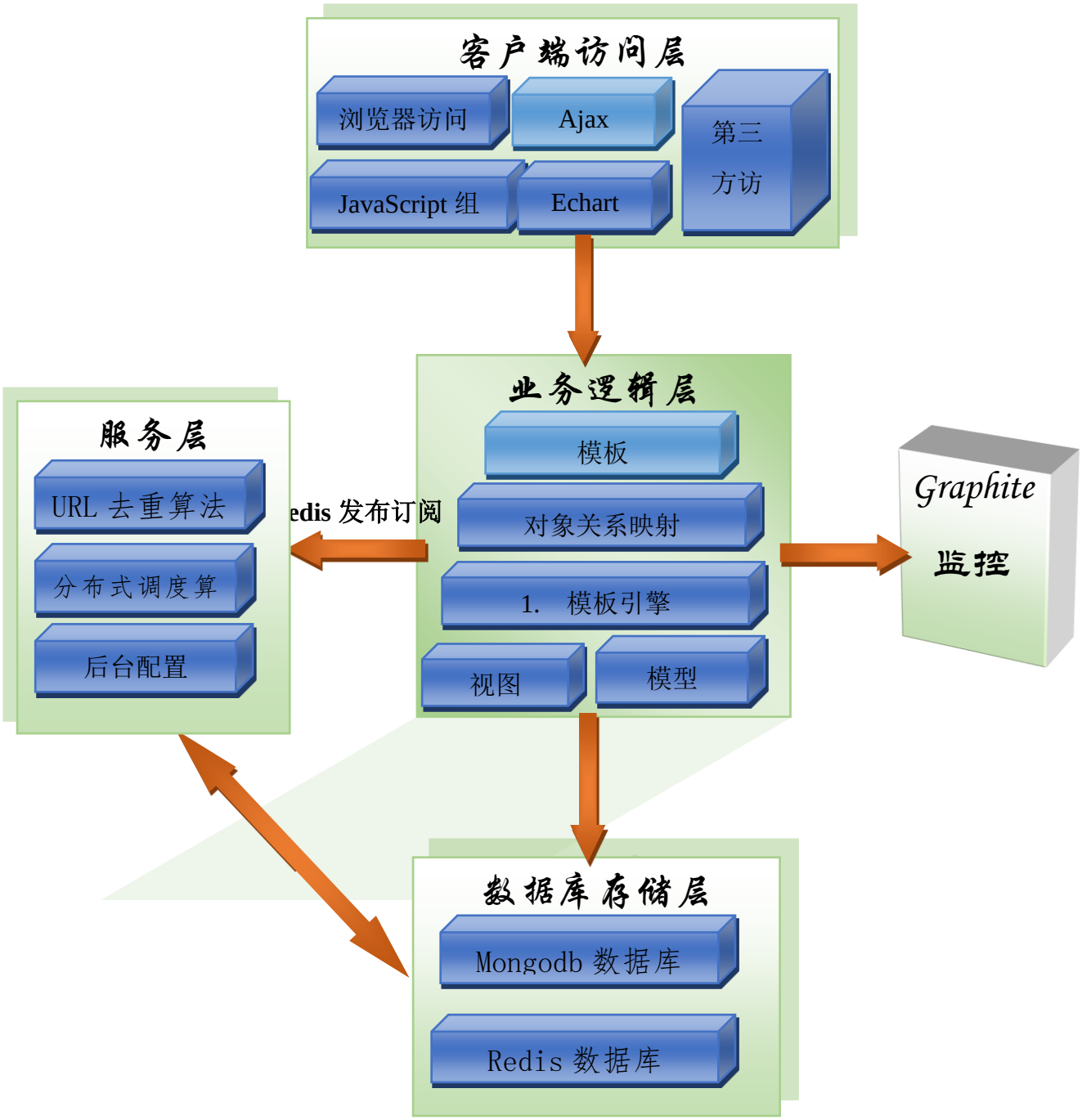
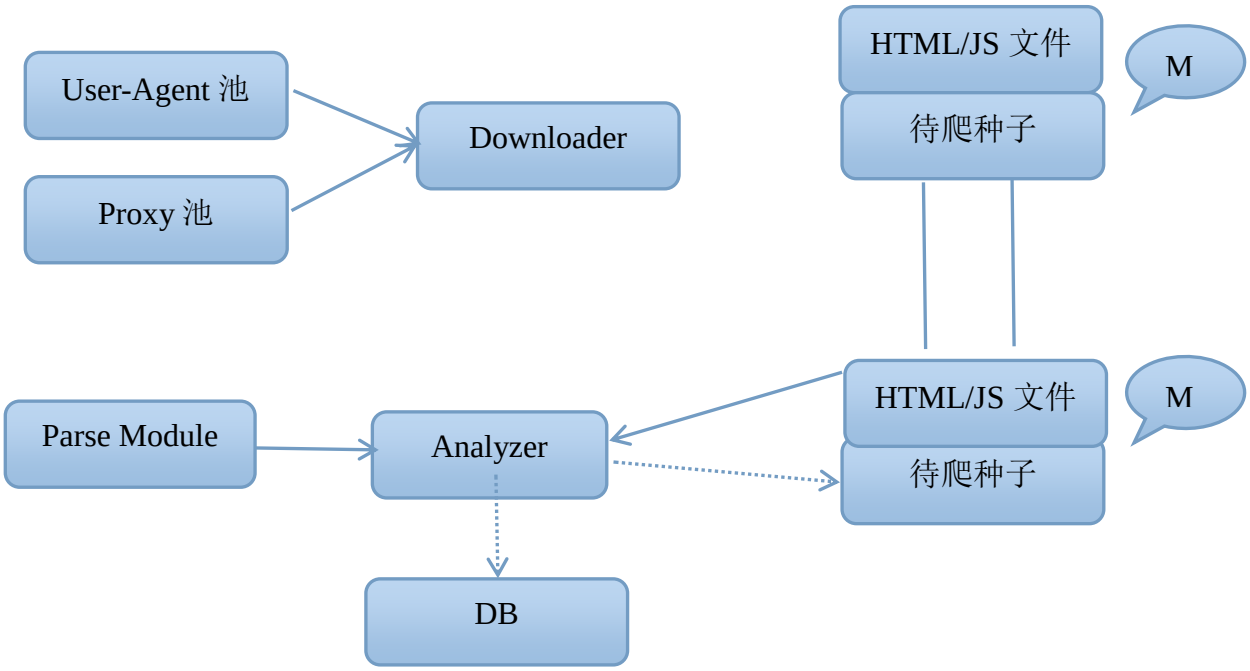


图 2-1 项目整体架构

该系统采用典型的分层架构。

- 客户端访问层负责处理用户的操作，接收用户的输入，并利用 GIS 客户端组件对服务层返回的内容进行处理；
- 业务逻辑层负责接收指定的参数并进行相应处理与计算，完成主要的业务功能，并负责数据的持久化；
- 服务层通过数据传输对象与业务逻辑层直接进行交互

- 数据层主要存储该系统所需要的数据。



架构设计思想

1) 框架主要分成两部分：下载器 Downloader 和解析器 Analyzer。Downloader 负责抓取网页，Analyzer 负责解析网页并入库。两者之间依靠消息队列 MQ 进行通信，两者可以分布在不同机器，也可分布在同一台机器。两者的数量也是灵活可变的，例如可能有三台机在做下载、两台机在做解析，这都是可以根据爬虫系统的状态及时调整的。

2) 从上图可以看到 MQ 有两个管道：HTML/JS 文件和待爬种子。Downloader 从待爬种子里拿到一条种子，根据种子信息调用相应的抓取模块进行网页抓取，然后存入 HTML/JS 文件这个通道；Analyzer 从 HTML/JS 文件里拿到一条网页内容，根据里面的信息调用相应的解析模块进行解析，将目标字段入库，需要的话还会解析出新的待爬种子加入 MQ。

3. 设计理念

该系统本着易使用，易拓展，健壮性易维护，易复用,分布式,爬虫优化,架构简明的设计理念，完成了整个课题的设计。下面做详细阐述。

3.1. 易使用

该系统有简洁的用户界面，方便用户操作与使用，系统设计的用户主界面如下：

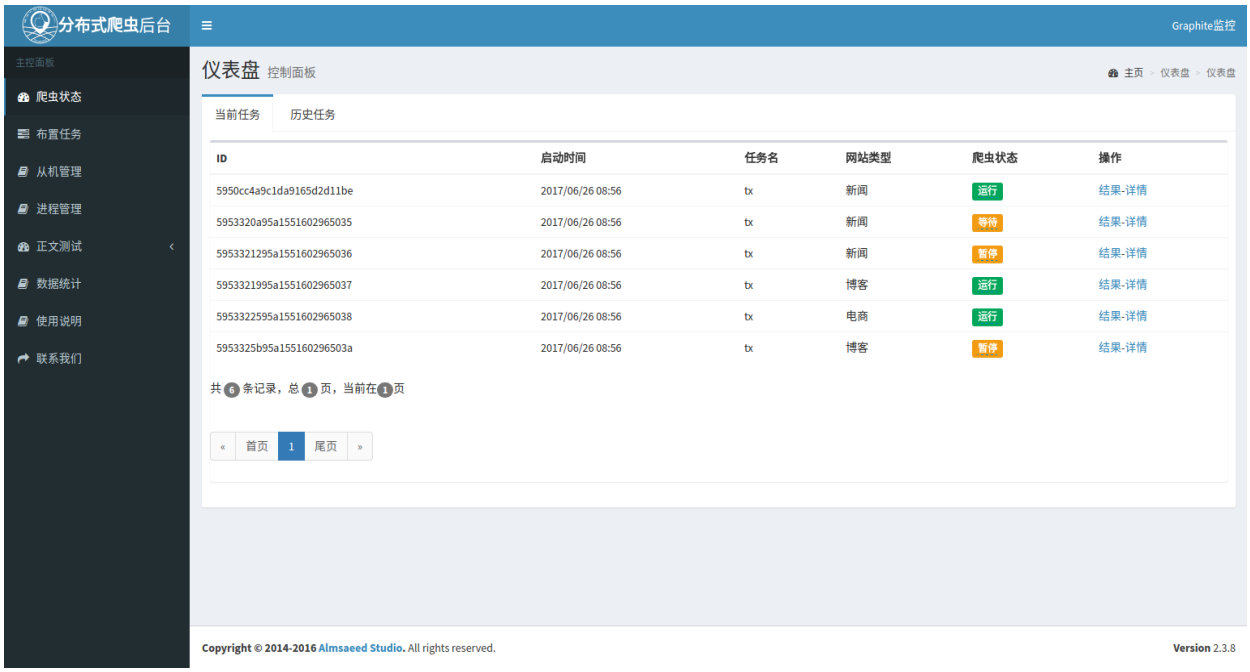


图 3.1-1 项目设计用主界面

用户在使用的时候，只需在界面中设置任务名,输入要爬取的网站 URL(s)；

3.2. 易拓展

多爬虫框架，这最直观的需求就是方便扩展，新增一个待爬的目标网站，我只需要写少量 必要的内容（如抓取规则、解析规则、入库规则）。该系统在设计时，始终秉承着“易拓展”的原则，使得系统更好地进行拓展和扩充，这里的拓展主要指的是：

- 横向拓展
- 纵向拓展

a) 横向拓展

横向拓展指的是系统可以方便地与其他系统进行集成，该系统后台使用 WebService 的方式，可以很方便地实现跨防火墙的异地调用，不仅可以为基于浏览器的客户端提供服务，任何与 Internet 建立连接的应用程序都可以向后台的 Web 服务发送 XML 格式的 SOAP 消息，而无需关心该应用程序采用何种语言何

种平台实现。达到了异构的程序基于标准协议的互相通信。

该系统可以方便地与环保部门或者相关政府，企业的业务进行紧密集成，真正达到代码的高复用。

b) 纵向拓展

纵向拓展是指系统可以比较方便地进行功能以及模块的扩充，而无须改变整个系统架构与设计，中间服务严格按照分层的原则，模块之间耦合度很低，日后可以方便地加入对其他网站爬取数据模块而无需改变现有模块的代码。

3.3. 健壮性易维护

这么多网站同时抓取，报错的概率更大，例如断网、中途被防爬、爬到“脏数据”等等。所以必须要做好日志监控，能实时监控爬虫系统的状态，能准确、详细地定位报错信息；另外要做好各种异常处理。该系统在以下方便达到了较高的可维护性:

可理解性

该系统内部结构以及相关实现易于理解，有详细的代码结构说明书以及整个项目的帮助文档帮助维护人员理解，整个编写过程采用结构化和面向对象的软件设计思想，并采用良好的代码写作规范，有助于提高系统的可理解性。

可测试性

该系统在设计阶段以及编程实现阶段就尽力地将程序设计成易诊断和易测试的，并且系统有完整的日志系统可供维护人员排查问题，日志记录在项目根目录下的 local.log 以及 ./Log/log.txt 中。

可修改性

该系统的的代码结构比较清晰，圈复杂度，继承深度较小，且类与类之间耦合度较低，易于后期的进一步修改和完善。

3.4. 易复用

代码复用，功能模块化。该系统的整体结构清晰，模块各自有其明确的功能职责划分，且对主要的功能点实现做了较好的封装，易于二次开发使用。如果针

对每个网站都写一个完整的爬虫，那其中必定包含了许多重复的工作，不仅开发效率不高，而且到后期整个爬虫项目会变得臃肿、难以管理。

3.5. 分布式

多网站抓取，数据量一般也比较大，可分布式扩展，这也是必需的功能了。分布式，需要注意做好消息队列，做好多结点统一去重。

3.6. 爬虫优化

这就是大话题了，但最基本的，框架应该要基于异步，或者使用协程+多进程。

3.7. 架构简明

方便以后未知功能模块的添加。

4. 总体设计

4.1. 系统运行环境

- 软件环境： 操作系统：linux
- 数据库：mongodb,redis
- 应用服务器：Ubuntu
- 硬件环境：若干台腾讯云服务器(ubuntu14,1 核 1GB 1Mbps 系统盘：云硬盘)
  一台笔记本(win10, CPU intel5200 显卡 AMD 275R)

4.2. 总体结构

整个系统的结构如下：

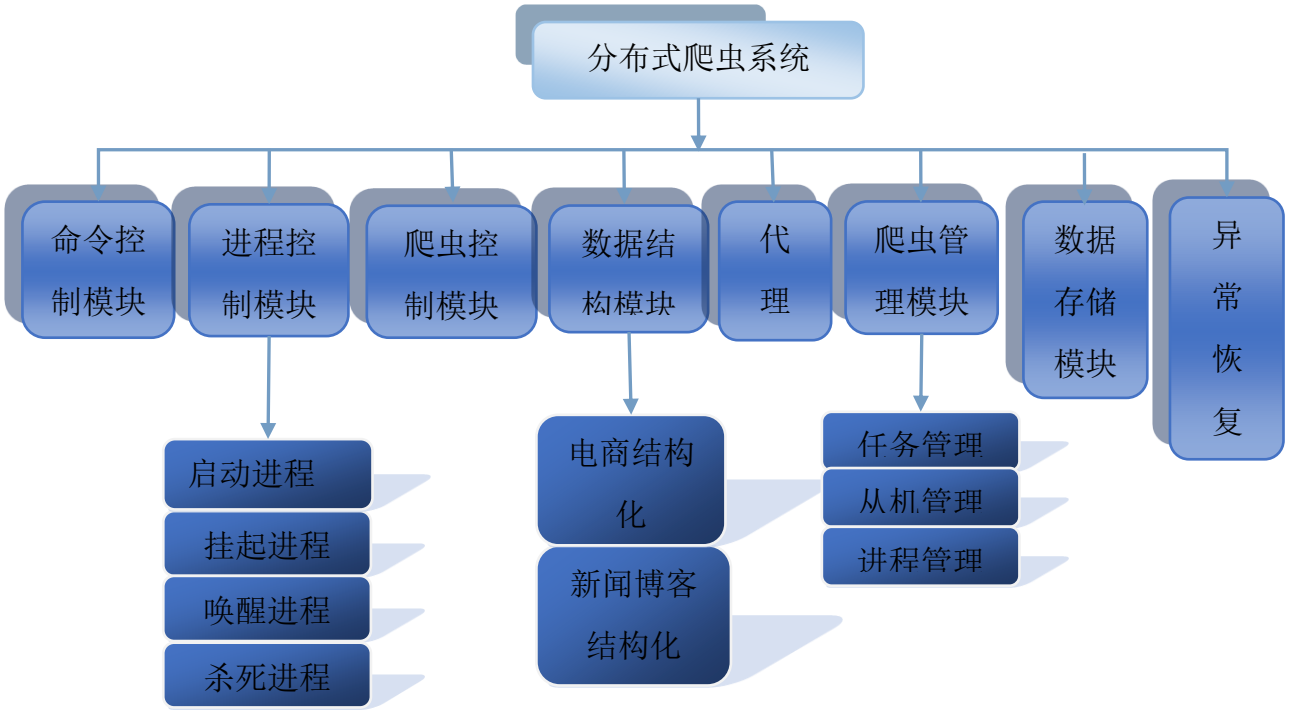


图 4.2-1 项目结构示意图

4.3. 技术路线

4.3.1. 系统开发环境

- Pycharm 2017
- Mongodb
- Python3.0

4.3.2. 开发技术

- Python 开发语言
- Django 模型
- Dom 解析技术:正则表达,BeautifulSoup ,xpath 等
- HTML+css+javascript 整合
- Scrapy,Scrapy-redis 框架运用
- Redis 与 Mongodb 非关系型数据库

4.4. 子系统清单

子系统编号	子系统名称	子系统英文名	子系统功能简述

		称	
WS_Common	公共模块	Common	对各个进程的唤醒,挂起
WS_TM	任务管理系统	Tasks Management	对各个爬虫任务的管理,选择开始、暂停、关闭等状态
WS_DM	分布式管理系统	Distributed Management	对分布式系统中的主从机进行管理,保持负载均衡
WS_Crawler	爬虫系统	Crawler	对网页数据进行自动化抓取

说明：子系统编号规则：系统名称_子系统名称，如系统或子系统名称过长，可采用英文简写。

4.5. 功能模块清单

模块编号	子系统名称	模块名称	模块功能简述	对应用例
WS_Crawler_1	爬虫系统	分布式协调服务	对其监控，一旦发现爬虫进程挂起，立即启动脚本对爬虫进程进行重新启动	启动爬虫
WS_Crawler_2		种子生成模块	根据当前页面自动获取 url 作为种子	爬取数据

WS_Crawler_3		网页采集模块	根据种子信息调用相应的抓取模块进行网页抓取	爬取数据
WS_Crawler_4		解析模块 BeautifulSoup&xPath	从 HTML/JS 文件里拿到一条网页内容，根据里面的信息调用相应的解析模块进行解析	爬取数据
WS_Crawler_5		Redis 数据库存储模块	临时存储待抓取的 URL	爬取数据

5. 模块设计

5.1. 模块说明

5.1.1. 任务管理模块

采集任务是指对一个或多个站点的页面的数据采集类型的归类，涉及采集的站点、采集种子信息、采集逻辑等相关信息，任务管理主要包括这些信息的管理和采集状态的管理，任务信息管理也称为任务编辑，就是将任务相关的信息建立成一些规范，而且是可由系统执行采集的基本单位。采集状态管理包括采集任务分发（由哪些采集终端执行），任务执行状态回收（任务是否执行成功），异常执行任务的识别及重试（为什么失败已经如何修复）等。

5.1.2. 资源管理模块

资源包括系统运行的硬件资源，也包括运行中需要的一些软件资源；硬件资源的管理包括服务器状态维护（比如有服务器挂起，终端就断开选择连接正常的

服务器)、采集终端状态管理（比如有终端挂起，服务器就不在分配任务到故障的采集终端)、网络线路管理（比如 ADSL 重拨）等。软件资源的管理包括 IP 资源（比如动态 IP 池）的管理、代理 IP 的管理、代理浏览器的管理等。这些硬件资源是用于执行采集系统，保障系统的正常运行；而软件资源主要辅助采集系统更好执行采集任务（比如解决网页下载的屏蔽问题)。

5.1.3. 内容下载模块

该功能主要涉及请求提交方式(GET、POST 等)，需要提交的内容，例如 URL、User-Agent、Referer URL、Cookie 等，各个站点，根据需要增加信息的信息。另外就是下载的实现方式，是选择已有的工具，比如 Webkit、HttpClient、URLConnection 等，还是自己通过 socket 等基础网络编程自己去实现。关于技术方面，还涉及到多线程或多进程的处理等。

5.1.4. 数据抽取模块

数据抽取在网络爬虫的研究中一直算是一个难点——如何在更少的人为干预下处理新资源，以及适应网站的改版等变化，提升系统的自适应能力。在大部分的实际应用中，自适应的技术使用的还是有限制的，这里就不涉及了。通常的数据抽取是通过下载内容的基本标准结构为基础来处理，比如 html、xml、pdf、excel 等，根据需求到指定的位置获取指定的内容信息，使用的技术比如正则表达式、xpath、BeautifulSoup 等。

5.1.5. 数据回收模块

数据回收指收集采集的内容并进行存储，采集的内容可以是抽取后的信息，也可以下载的原始网页、文档等。这里可能涉及到数据库技术，具体的要根据实际使用来选取。

5.2. 模块功能设计

5.2.1. 后台功能设计(Scray 框架)

后台服务的主要功能有：

- 模型的计算（该系统中主要包括 Django 模型）

- 下载器 Downloader
- 解析器 Analyzer

下面说明各功能的详细设计。

5.2.1.1. 模型的计算

该系统中主要用到的模型有：

- Django 模型

在 Django 的框架设计中采用了 mtv 模型，即 Model,template,viewer. Model 相对于传统的三层或者 mvc 框架来说就相当对数据处理层，它主要负责与数据的交互，在使用 django 框架设计应用系统时，需要注意的是 django 默认采用的是 orm 框架中的 codefirst 模型，也就是说开发人员只需要专注于代码的编写，而不需要过多的关注数据库层面的东西，把开发人员从数据库中解放出来.

Django 会根据 Model 类生成一个数据库镜像文件，然后再使用该镜像文件生成数据库，同时该文件将记录与数据库同步版本的变化，所以在使用 django 进行开发时不要手工去修改数据库，这样会造成 django 框架的版本记录不正确，从而无法正确的同步数据模型与数据库的内容.镜像文件内容记录了对 model 所在的 app 的记录，以及执行的动作.

其四部曲:

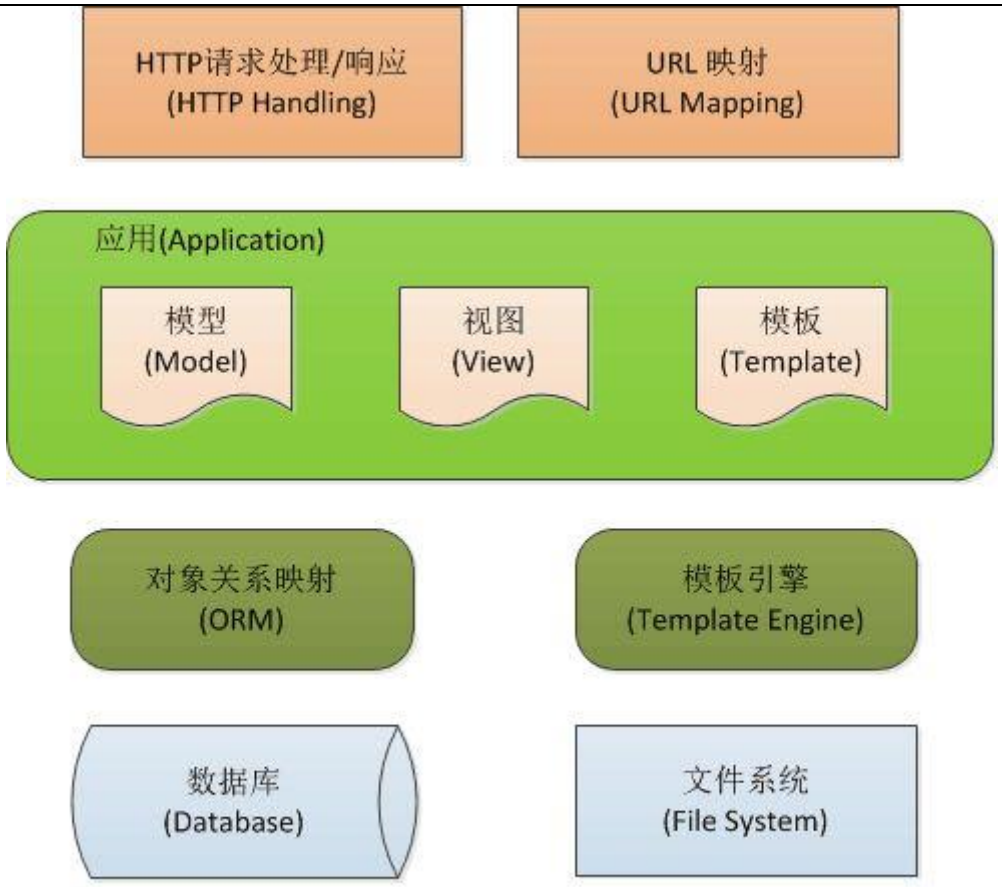
1)配置数据库

2)App 注册到工程

3)编写 model

4)执行命令同步

对 model 做调整后只需要重复执行第四步即可同步数据库



Django架构总览图

5.2.1.2. 下载器 Downloader

Downloader 是包含 User-Agent 池、Proxy 池的,适合复杂网站的抓取.首先从初始 URL 开始，调度(Scheduler)会将其交给下载器(Downloader)进行下载。

Scrapy 中的数据流由执行引擎控制。

1)引擎打开一个网站，找到处理该网站的 Spider 并向该 spider 请求第一个爬取的 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。

5.2.1.3. 解析器 Analyzer

下载之后会交给蜘蛛（Spider）进行分析,蜘蛛（Spider)分析出来的结果有两种：一种是需要进一步抓取的链接，例如之前分析的“下一页”的链接，这些东西会被传回调度（Scheduler)；另一种是需要保存的数据，它们则被送到项目管道(Item Pipeline)那里，那是对数据进行后期处理（详细分析、过滤、存储等）的地方。

多个解析器 Analyzer 共用一个去重队列，才能够保证数据的统一不重复。去重队列可以放在一台机上。基于 Redis 实现了 Bloomfilter 算法。

1)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。

2)Spider 处理 Response 并返回爬取到的 Item 及(跟进的)新的 Request 给引擎。

3)引擎将(Spider 返回的)爬取到的 Item 给项目管道(Item Pipeline)，将(Spider 返回的)Request 给调度器。

4)(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。

性能

灵活性：容错能力较强，响应时间短（5s 以内）。

5.3. 故障处理说明

出错名称	系统输出信息	处理方法
用户名输入错误	“用户名错！请重新输入！”	进入登录页面

用户输入密码错误	“您的密码输入有误，请重新输入！”	进入登录页面
系统故障	“服务器维护中！暂停服务！”	立即启用备用机，恢复故障
参数未输入完全	对应位置闪烁	等待用户输入

6. 数据库设计

该系统使用的是 nosql 非关系型数据库 mongodb 和 Redis。mongodb 是一个基于分布式文件存储的数据库，其模式自由（schema-free），意味着对于存储在 mongodb 数据库中的文件，我们不需要知道它的任何结构定义。如果需要的话，你完全可以把不同结构的文件存储在同一个数据库里。根据本系统信息量多且种类繁多，该数据库较为适用存储爬取过来的信息。Redis 是分布式的 Key-Value 数据库，适用于存储临时的页面 URL 的仓库。

6.1. 数据库概念模型

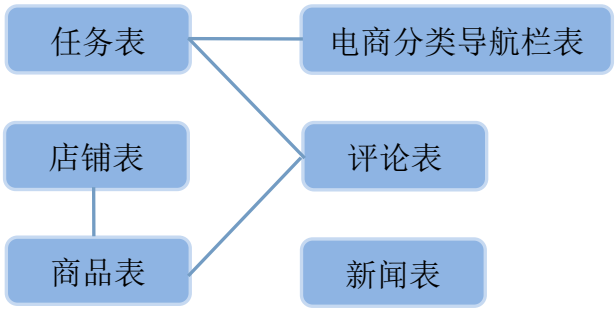


图 7.1-1 数据库概念模型

数据库概念模型如上所示，其中任务与电商分类导航栏是一对多的关系，任务与店铺是一对一的关系，任务与评论一对一的关系，任务与新闻一对一的关系，店铺与商品一对一的关系，商品与评论一对一的关系。

6.2. 数据库物理模型

图 7.2-1 数据库物理模型

6.3. 数据库表设计

1) 任务表

摘要	字段编码	字段名	数据类型	说明	是否可空
既定类型（代码编写时或已经出现的字段。）	TaskID	编号（自动生成）	Object		否
	TaskStartTime	开始时间	String		否
	TaskEndTime	结束时间	String		否
	TaskDSA	分布式从机地址（如有多个,分割）	String	例如 115.125.122.1,11.125.11.22 或域名	否
	TaskType	任务类型	String	电商、新闻、博客等	否
	TaskURLs	任务 URLs（如有多个,分割）	String	https://www.taobao.com, https://www.jd.com	否
动态字段（支持动态扩展，或删减）		可自动扩展字段			是
					是

2）电商分类导航栏表

摘要	字段编码	字段名	数据类型	说明	是否可空
既定类型	ElectricityID	编号（自动生成）	Object		否
	TaskID	所属任务编号	Object	虚拟外键，连接到任务表	否

	ElectricityCN	分类名	String	类别名，例如手机	否
	ElectricityCURL	分类 URL	String	该条目的 URL	否
	ElectricityNavigation	导航级别	String	用以鉴别该条目在整个导航中的位置存为 1,11,111 等	否
扩展字段		可自动扩展字段			是
		...			是
补充	域名可用 url 正则出				

3)店铺表

4)商品表

摘要	字段编码	字段名	数据类型	说明	是否可空
既定类型	ShopID	编号（自动生成）	Object		否
	TaskID	所属任务编号	Object	虚拟外键，连接到任务	否
	ShopURL	店铺 URL	String	店铺 URL	否
	ShopName	店铺名	String	店铺名称	否
	ShopGoodsNum	店铺商品数	String	店铺中的商品数目	否
	ShopRating	店铺评价等	String	不需要具体评论，只需要优良差，或几点几分	否
动态字段		可自动扩展字段			是
		...			是

摘要	字段编码	字段名	数据类型	说明	是否可空
既定类型	CommodityID	编号（自动生成）	Object		否
	TaskID	所属任务编号	Object		否
	ShopID	所属店铺编号	Object	虚拟外键，连接到店铺表	否
	CommodityURL	商品 URL	String	商品的 URL	否
	CommodityName	商品名	String	店铺中的商品数目	否
	CommodityPrice	商品价格	String	商品原价	否
	CommoditySalePrice	促销价	String	商品的促销价	否
	CommodityRating	商品评等	String	不需要具体评论，只需要优良差，或几点几分	否
动态字段		可自动扩展字段			是
		...			是
补充	商品评论，直接查询评价表获取，与商品的评等有所不同				

5）进程表

摘要	字段编码	字段名	数据类型	说明	是否可空
既定类型	ID	编号（自动生成）	Object		否
	TaskID	所属任务编号	Object		否
	processID	进程编号	Object		否

	IP	主机 IP	String		否

6)新闻表

摘要	字段编码	字段名	数据类型	说明	是否可空
既定类型	NewsID	编号（自动生成）	Object		否
	TaskID	任务编号	String	虚拟外键，连接到任务编号	否
	NewsTime	发布时间	String		否
	NewsTitile	标题	String		否
	NewsKeywords	关键字	String		否
	NewsContent	内容	String		否
动态字段	NewsClassfication	分类			是
		...			是

7. 可重用子系统或模块

可重用模块	应用范围
分布式协调服务	需要监控进程,保存进程挂起后能立即重新启动
种子生成模块	需要自动获取 URL 的地方
解析模块	需要对海量数据进行解析的地方
Redis 数据库存储模块	需要用到 Redis 数据库存储的地方

8. 核心算法与技术攻关

8.1. 分布式调度算法

分布式爬虫将所有任务在多台机器上分布式执行（可用多进程模拟）。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。

调度算法应该在满足特定的条件下，实现最大的下载量。

8.2. URL 去重算法

对 URL 进行去重，已经下载过的，没有进行数据更新的，不再进行下载。去重算法应考虑内存的问题，内存越小越优。即,去重只需要考虑两个点：去重的数据量、去重速度。

8.2.1. 去重与数据量

1)数据量不大时，可以直接放在内存里面进行去重，例如 python 可以使用 set()进行去重。

2)当去重数据需要持久化时可以使用 redis 的 set 数据结构。

3)当数据量再大一点时，可以用不同的加密算法先将长字符串压缩成 16/32/40 个字符，再使用上面两种方法去重；

4)当数据量达到亿（甚至十亿、百亿）数量级时，内存有限，必须用“位”来去重，才能够满足需求。Bloomfilter 就是将去重对象映射到几个内存“位”，通过几个位的 0/1 值来判断一个对象是否已经存在。

5)然而 Bloomfilter 运行在一台机器的内存上，不方便持久化（机器 down 掉就什么都没啦），也不方便分布式爬虫的统一去重。如果可以在 Redis 上申请内存进行 Bloomfilter，以上两个问题就都能解决了。

8.2.2. Bloomfilter 算法去重

1)Bloomfilter 算法如何使用位去重，简单点说就是有几个 seeds，现在申请一段内存空间，一个 seed 可以和字符串哈希映射到这段内存上的一个位，

几个位都为 1 即表示该字符串已经存在。插入的时候也是，将映射出的几个位都置为 1。

2)Bloomfilter 算法会有漏失概率，即不存在的字符串有一定概率被误判为已经存在。这个概率的大小与 seeds 的数量、申请的内存大小、去重对象的数量有关。下面有一张表，m 表示内存大小（多少个位），n 表示去重对象的数量，k 表示 seed 的个数。例如我代码中申请了 256M，即 1<<31（m=2^31，约 21.5 亿），seed 设置了 7 个。看 k=7 那一列，当漏失率为 8.56e-05 时，m/n 值为 23。所以 n = 21.5/23 = 0.93(亿)，表示漏失概率为 8.56e-05 时，256M 内存可满足 0.93 亿条字符串的去重。同理当漏失率 0.000112 时，256M 内存可满足 0.98 亿条字符串的去重。

m/n	k	k=1	k=2	k=3	k=4	k=5	k=6	k=7	k=8
2	1.39	0.393	0.400						
3	2.08	0.283	0.237	0.253					
4	2.77	0.221	0.155	0.147	0.160				
5	3.46	0.181	0.109	0.092	0.092	0.101			
6	4.16	0.154	0.0804	0.0609	0.0561	0.0578	0.0638		
7	4.85	0.133	0.0618	0.0423	0.0359	0.0347	0.0364		
8	5.55	0.118	0.0489	0.0306	0.024	0.0217	0.0216	0.0229	
9	6.24	0.105	0.0397	0.0228	0.0166	0.0141	0.0133	0.0135	0.0145
10	6.93	0.0952	0.0329	0.0174	0.0118	0.00943	0.00844	0.00819	0.00846
11	7.62	0.0869	0.0276	0.0136	0.00864	0.0065	0.00552	0.00513	0.00509
12	8.32	0.08	0.0236	0.0108	0.00646	0.00459	0.00371	0.00329	0.00314
13	9.01	0.074	0.0203	0.00875	0.00492	0.00332	0.00255	0.00217	0.00199
14	9.7	0.0689	0.0177	0.00718	0.00381	0.00244	0.00179	0.00146	0.00129
15	10.4	0.0645	0.0156	0.00596	0.003	0.00183	0.00128	0.001	0.000852
16	11.1	0.0606	0.0138	0.005	0.00239	0.00139	0.000935	0.000702	0.000574
17	11.8	0.0571	0.0123	0.00423	0.00193	0.00107	0.000692	0.000499	0.000394
18	12.5	0.054	0.0111	0.00362	0.00158	0.000839	0.000519	0.00036	0.000275
19	13.2	0.0513	0.00998	0.00312	0.0013	0.000663	0.000394	0.000264	0.000194
20	13.9	0.0488	0.00906	0.0027	0.00108	0.00053	0.000303	0.000196	0.00014
21	14.6	0.0465	0.00825	0.00236	0.000905	0.000427	0.000236	0.000147	0.000101
22	15.2	0.0444	0.00755	0.00207	0.000764	0.000347	0.000185	0.000112	7.46e-05
23	15.9	0.0425	0.00694	0.00183	0.000649	0.000285	0.000147	8.56e-05	5.55e-05
24	16.6	0.0408	0.00639	0.00162	0.000555	0.000235	0.000117	6.63e-05	4.17e-05
25	17.3	0.0392	0.00591	0.00145	0.000478	0.000196	9.44e-05	5.18e-05	3.16e-05
26	18	0.0377	0.00548	0.00129	0.000413	0.000164	7.66e-05	4.08e-05	2.42e-05
27	18.7	0.0364	0.0051	0.00116	0.000359	0.000138	6.26e-05	3.24e-05	1.87e-05
28	19.4	0.0351	0.00475	0.00105	0.000314	0.000117	5.15e-05	2.59e-05	1.46e-05
29	20.1	0.0339	0.00444	0.000949	0.000276	9.96e-05	4.26e-05	2.09e-05	1.14e-05
30	20.8	0.0328	0.00416	0.000862	0.000243	8.53e-05	3.55e-05	1.69e-05	9.01e-06
31	21.5	0.0317	0.0039	0.000785	0.000215	7.33e-05	2.97e-05	1.38e-05	7.16e-06
32	22.2	0.0308	0.00367	0.000717	0.000191	6.33e-05	2.5e-05	1.13e-05	5.73e-06

3)基于 Redis 的 Bloomfilter 去重，其实就是利用了 Redis 的 String 数据结构，但 Redis 一个 String 最大只能 512M，所以如果去重的数据量大，需要申请多个去重块（代码中 blockNum 即表示去重块的数量）。

4)代码中使用了 MD5 加密压缩，将字符串压缩到了 32 个字符（也可用 hashlib.sha1()压缩成 40 个字符）。它有两个作用，一是 Bloomfilter 对一个很长的字符串哈希映射的时候会出错，经常误判为已存在，压缩后就不再有这个问题；二是压缩后的字符为 0~f 共 16 中可能，我截取了前两个字符，再根据 blockNum 将字符串指定到不同的去重块进行去重。

8.3. Dom 自动结构化

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。