西安邮电大学
毕业论文

题目： 基于 Python 网络爬虫的天气信息搜索与预报系统

学院： 通信与信息工程学院

专业： 通信工程

班级： 通工 1310

学生姓名： 鲍颖

学号： 03131313

导师姓名： 石薇 职称： 讲师

起止时间：2017 年 02 月 27 日至 2017 年 06 月 17 日

毕业设计（论文）声明书

本人所提交的毕业论文《基于Python网络爬虫的天气信息搜索与预报系统》是本人在指导教师指导下独立研究、写作的成果，论文中所引用他人的文献、数据、图件、资料均已明确标注；对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式注明并表示感谢。

本人完全理解《西安邮电大学本科毕业设计（论文）管理办法》的各项规定并自愿遵守。

本人深知本声明书的法律责任，违规后果由本人承担。

论文作者签名：

日期：　　年　月　日

西安邮电大学本科毕业设计(论文)选题审批表

<table>
<tr><td>申报人</td><td colspan="2">石薇</td><td>职称</td><td>讲师</td><td>学院</td><td colspan="2">通信与信息工程学院</td></tr>
<tr><td>题目名称</td><td colspan="7">基于 Python 网络爬虫的天气信息搜索与预报系统</td></tr>
<tr><td>题目来源</td><td>科研</td><td colspan="2"></td><td colspan="2">教学</td><td>其它</td><td>√</td></tr>
<tr><td>题目类型</td><td>硬件设计</td><td></td><td>软件设计</td><td>√</td><td>论文</td><td>艺术作品</td><td></td></tr>
<tr><td>题目性质</td><td colspan="2">应用研究</td><td colspan="2">√</td><td colspan="2">理论研究</td><td></td></tr>
<tr><td>题目简述</td><td colspan="7">爬虫程序(Spider)是一种能够自动搜集互联网信息的程序，通过网络爬虫不仅能够为搜索引擎采集网络信息，而且可以作为定向信息采集器，定向采集某些网站下的特定信息，如招聘信息，租房信息等。本题采用基于 Python 语言与 scrapy 框架实现的爬虫程序，从网络上获取包含天气信息的网页，使用多种工具解析并获取其中的信息并进行分类存储，在客户端程序中，可以获取到用户所需要的目的城市的天气信息。可以实时更新当前天气状况。</td></tr>
<tr><td>对学生知识与能力要求</td><td colspan="7">1.对 Mysql 数据库的熟练应用 2.对 Python 开发的熟练应用 3.对 scrapy 框架的掌握 4.使用 PyQt 进行 GUI 界面的设计与开发 5.使用 request+beautifsoup 或进行 html 网页的获取与内容解析 6.开发多线程程序的能力 7.对采集到的数据进行科学的处理与分析 7.端正的学习态度和一定的自学能力</td></tr>
<tr><td>具体任务以及预期目标</td><td colspan="7">本课题要求学生运用 Python+PyQt 开发集成环境,数据库采用 Mysql,使用 scrapy 框架来进行底层爬虫程序的设计。此程序应包含一个长期运行的服务器程序，与一个可以查看信息的客户端程序。本课题的成果形式为：网络爬虫天气预报程序服务端+客户端程序。爬虫程序可以定时获取用户所定制的城市的天气情况，并随时可以修改城市。用户可与获得实时的天气情况，与未来天气预报。</td></tr>
<tr><td>时间进度</td><td colspan="7">2017 年 2 月 27 日至 2017 年 3 月 05 日: 1. 熟悉 Mysql 数据库，PyQt 以及 scrapy 框架，熟悉 Python 语法；2. 2017 年 3 月 06 日至 2016 年 3 月 23 日：进行程序整体结构的组织，学习与优化数据的处理策略；3. 2017 年 3 月 24 日至 2017 年 5 月 20 日：编码实现此系统；4. 2017 年 5 月 21 日至 2017 年 6 月 17 日：撰写论文，准备答辩。</td></tr>
<tr><td colspan="2">系（教研室）主任签字</td><td colspan="2">年 月 日</td><td>主管院长签字</td><td colspan="3">年 月 日</td></tr>
</table>

西安邮电大学本科毕业设计（论文）开题报告

学生姓名	鲍颖	学号	03131313	专业班级	通工 1310
指导教师	石薇	题目	基于 Python 网络爬虫的天气信息搜索与预报系统		

选题目的（为什么选该课题）

互联网是一个非常庞大的数据库，各种各样的数据与资源以各种形式存储在互联网上，在网页展示形式也非常多：文字、图片、Flash 动画或者视频等等。当用户试图在网页上获取一些信息并点击链接的时候，往往会收到大量外带的数据，例如广告推送、网页浮动窗口等，不但容易干扰视线，使用户较难筛选出有效信息，而且某些广告弹窗可能会暗藏一些恶意链接，使用户造成不必要的损失。本课题的主要目的是设计一个可以主动定向收集天气信息的网络爬虫程序，使用户试图去在网络上获取信息的时候，避开这些可能存在的“坑”，高效、快速的完成信息的获取，此程序还可支持用户进行指定城市的天气信息的订阅，被订阅的城市近期的天气数据会主动推送到用户端进行显示，并可根据历史数据，得到每日的降水率，进行一定程度的天气预测。

前期基础（已学课程、掌握的工具，资料积累、软硬件条件等）

到目前我已经学习了数据结构、计算机操作系统、网络编程等课程，掌握了 mysql 数据库、VS2015、Qt 等软件工具的使用方法，可以熟练使用的计算机语言有 Python/Shell、C++，具备了开发 linux 服务器端程序的能力，已经查阅的资料包括《Python 核心编程》(美) 丘恩 (Chun, W.J.) 人民邮电出版社、《Unix 网络编程》(美) 史蒂文斯 人民邮电出版社等。软件条件包括 windows/Linux 操作系统、VS2015、Mysql、PyQt 等，硬件条件包括个人开发机器一台(内存 8G,CPU:i5-4200H)，阿里云服务器一台(内存 1G,单核 CPU,带宽 100M)。

要研究和解决的问题（做什么）

1.开发环境的搭建，包括：Python 环境的搭建以及 PyQt、VS2017、Mysql 等软件的安装。 2.熟练掌握 Python 编程以及 Linux 操作系统的常用命令。 3.了解网络爬虫技术研究现状，学习网络爬虫的技术原理。 4.学习已经有的爬虫开源框架，了解其优点与缺点，结合自己的使用场景进行优化。 5.设计数据的存储格式，使用什么数据结构。 6.设计与选择客户端与服务器端之间的数据传输协议与格式。 7.对所爬取网站的主体内容与结构的分析。 8.若网站有防爬策略，比如 IP 限频、IP 黑名单等，该如何解决。 9.对网络爬虫工具性能进行测试，对程序功能进行完善。 10.对所爬取到的数据进行分析与归类，让推送到客户端的数据更准确。 11.根据历史数据进行一定程度上的天气分析与预测。

工作思路和方案（怎么做）

我的工作思路：软件开发中，搭建一个开发环境是最先需要完成的任务，开发环境包括 windows 上的 VS2015、PyQt，Linux 上的 Mysql、Python3 以及一些软件的环境变量的配置等。之后需要进行一些相关资料与知识的搜索与积累，例如 windows 下使用 VS2015 和 PyQt 进行 GUI 开发的技术，linux 下的网络编程的知识，和网络爬虫程序相关的开源框架 scrapy 的原理的学习与研究，以及如何使用 python 第三方库 request+beautifulsoup 进行 html 网页内容的获取与解析等，预计学习时间为 2017 年 2 月 27 日至 2017 年 3 月 15 日。之后需要开始进行程序的开发，设计与实现网络爬虫程序，并将其作为一个长期运行的服务器程序进行开发，其中编写爬虫程序需要使用 scrapy 框架，网页内容的请求与获取需要使用第三方库，对所获取网页的内容的解析与信息的提取需要用到正则表达式等技术 。服务器程序要能够在指定时间完成数据的更新功能，这需要使用到到一些 Linux 下命令行操作与 Shell 脚本编程的技术，且由于是先进行服务器程序的开发，此时尚未有客户端程序，所以需要在代码中加入调试信息，观察每一个步骤获取到的数据以对服务器程序进行调试，编写测试程序模拟客户端对服务器发起请求，以测试服务器的性能，服务器程序开发时间预计 2017 年 3 月 16 日到 2017 年 4 月 25 日。之后进行客户端程序的开发，此环节较服务器程序开发较为简单，使用 PyQt 进行 GUI 界面的开发，并向服务器发起请求，取回已经被服务器分析、处理过的数据，预计开发时间 2017 年 4 月 26 日到 2017 年 5 月 10 日。撰写论文，完成论文初稿，2017 年 5 月 15 日至 2017 年 5 月 30 日。完善并修改毕业论文，2017 年 6 月 1 日至 2017 年 6 月 7 日。 最后准备答辩，2017 年 6 月 8 日至 2017 年 6 月 17 日。

指导教师意见

签字　　年　月　日

西安邮电大学毕业设计（论文）成绩评定表

<table>
<tr><td>学生姓名</td><td>鲍颖</td><td>性别</td><td>男</td><td>学号</td><td>03131313</td><td>专业
班级</td><td>通工 1310</td></tr>
<tr><td>课题名称</td><td colspan="7">基于Python网络爬虫的天气信息搜索与预报系统</td></tr>
<tr><td>指导
教师
意见</td><td colspan="7">评分（百分制）：　　指导教师(签字)：　　年　　月　　日</td></tr>
<tr><td>评阅
（验收）
意见</td><td colspan="7">评分（百分制）：　　评阅教师(签字)：　　年　　月　　日</td></tr>
<tr><td>答辩
小组
意见</td><td colspan="7">评分（百分制）：　　答辩小组组长(签字)：　　年　　月　　日</td></tr>
<tr><td>评分比例</td><td colspan="7">指导教师评分　（%）评阅（验收）评分　　（%）　答辩小组评分　（%）</td></tr>
<tr><td>学生总评
成绩</td><td colspan="2">百分制成绩</td><td></td><td colspan="2">等级制成绩</td><td colspan="2"></td></tr>
<tr><td>答辩委员
会意见</td><td colspan="7">毕业论文(设计)最终成绩(等级)：

学院答辩委员会主任(签字、学院盖章)：　　年　　月　　日</td></tr>
</table>

目 录

摘　　要

在互联网发展的初期，用户通过访问知名门户站点的方式搜集自己想要的数据信息，但是随着互联网的发展与扩大，其中汇聚的数据量正在以指数级别爆炸式的增长。海量般的数据对用户精确搜集信息的需求造成了困扰。因此，通过网络爬虫技术来进行高效、快速的数据搜集，将可以给用户提供一个纯净、可靠地获取信息的方式。本论文提出了一种可以帮助用户自动收集未来天气信息的软件设计理念，并通过网络爬虫技术、网络编程技术、GUI 客户端开发技术对这个设计理念进行了实现。本论文先对网络爬虫的软件结构进行了分析，然后介绍了一些开源的优秀的网络爬虫的框架，随后重点介绍了网络爬虫的搜索策略与网络编程的一些技术概念。通过 Scrapy 框架与 python 语言对了这个网络爬虫程序进行了实现，并使用 Qt 与 C++语言对 GUI 客户端进行实现。通过 C/S 软件设计模型，将爬虫程序和客户端程序连接起来，构建成一套结构完整的工具软件。用户在使用本软件在去获取信息的时候，能够避开大量无效信息的干扰，使用户能够高效、快速的完成天气信息的获取。

关键词：网络爬虫；python；客户端/服务器模式；Qt；网络编程

ABSTRACT

In the early stages of Internet development, users access the well-known portal site to collect the information they want, but with the development and expansion of the Internet, which aggregated data volume is the explosive growth of the exponential level. Massive data on the user to accurately gather information needs caused trouble. Therefore, through the web crawler technology to carry out efficient and fast data collection, will be able to give users a pure, reliable way to obtain information. This paper presents a software design concept which can help users to collect future weather information automatically. The design concept is realized through network reptile technology, network programming technology and GUI client development technology. This paper first analyzes the software structure of the web crawler, and then introduces some excellent open source crawler framework, and then introduces some of the technical concepts of network crawler search strategy and network programming. This web crawler is implemented through the Scrapy framework and python language, and the GUI client is implemented using Qt and C ++ languages. Through the C / S software design model, the reptile program and the client program to connect together to build a complete set of structural tools. Users in the use of the software to get information, can avoid a large number of invalid information interference, so that users can efficiently and quickly complete the weather information access.

Key words：Web Crawler；python；Client/server mode；Qt；Network programming

第一章　绪论

1.1 选题背景与意义

随着时代的进步与科技的发展，互联网已经在不知不觉间蔓延、扩散到了我们生活中的每一个角落。近年来被社会谈论的非常热门的一些科技词汇，比如大数据、云计算、物联网等，几乎都和互联网有很强的关联性。互联网具有的多维性、自由平等性、虚拟交互性、以及最重要的海量性与全球性，改变了生产信息与传播信息的方式，进而改变了人类的工作方式、生活方式、生产方式甚至是思维方式。人们已逐渐习惯通过互联网去获取自己需要的各种信息（浏览新闻网站或者在线观看视频），或者是通过互联网去发布一些自己愿意公开的信息（微博、说说），而不是通过那些传统渠道（报纸、杂志），传统渠道在各方面所占的份额也在逐渐降低，这证明人们获取与发布信息的方式正在转变。

然而这种情景的背后，有我们需要迫切关注的一点，那就是每天都有 EB 级别的新数据涌入互联网这个庞大的数据库，那么随着数据的不断累积，摆在我们家面前的的就是一个拥有海量数据的无比庞大的数据库，各种各样的信息全部沉淀、堆积在这个庞大的数据库之中。这就意味着，当用户想要从互联网上接收信息的时候，不一定能够快速、准确的获取到自己真正想要的信息，而且这些信息中或许掺杂了非常多的无用数据，甚至于是包含了恶意链接的、会造成用户不必要损失的内容。因此，能够从网络中定向的、准确的、高效的提取用户需要的信息，才能提升其使用体验。本课题以用户使用频率较高的天气搜索功能为起点，通过网络爬虫技术与 C/S 软件架构模型，设计并实现了一套能够高效获取互联网天气数据并分析的的软件，可以精确、快速的获取天气信息，降低用户浪费在数据分辨与筛选上的时间，为用户提供一个纯净、可靠的信息获取来源。

1.2 网络爬虫技术发展现状

传统的网络爬虫技术想要解决的问题范围主要是静态 web 页面。基于 socket 的 httpclient 功能简单，性能强大，特别是在多线程、高并发的情境下，从而被开发者青睐。特别是搜索引擎中，若抓取的是静态页面，httpclient 是非常适合这种情景的。随着 AJAX 技术的兴起，很多网页通过 AJAX 技术改写为动态网页。如何抓取动态页面，就成了现代网络爬虫需要解决的问题。当网络爬虫遇到 AJAX 加载的信息、需要浏览器内核使用 JS 进行渲染的时候，就无法正常工作了。所以若爬虫程序还是按照传统的下载网页、分析文档、提取链接的机制的话，就抓取不到动态页面尚未加载出来的数据。传统的爬虫采用的是 xpath 或正则表达式这种技术来进行内容的提取，无法理解 JS 的语法，所以传统的爬虫技术是无法模拟出触发 JS 的异步调用并解析返回的异步回调逻辑和内容。

若爬虫程序的可行性得到了解决，那么其面临的最大问题就是程序性能的提

升和突破网站对爬虫的封锁。大部分情况下，采用多线程高并发、高频率进行数据抓取是可行的，但是这有一个前提，那就是网站没有采用任何反爬虫措施，例如验证码、防火墙、限制访问频率等等。但更多的时候，真正有价值的信息，一定是伴随着严格的反爬取策略的，一旦网站检测出当前请求者是爬虫程序而非真正的用户，那么最严格的情况下，网站可能会封掉正在运行爬虫程序的机器的 IP 地址，也就是加入黑名单中。一旦被 IP 地址被加入黑名单，就意味着当前运行爬虫程序的服务器再也无法访问这个网站，更别提爬取数据了。所以，由此诞生出很多反封锁的办法，例如维护一个代理 IP 池来防止网站封 IP，但是这也带来了新的问题，那就是 IP 池是否稳定，和切换 IP 的速度等等，这些都是开发出一款稳定的、功能完善的网络爬虫而无法避免、需要考虑的问题。

1.3 本文章节安排

本文章节具体安排如下：

第一章：引言部分。主要用于介绍背景知识，包括网络爬虫技术发展现状，以及客户端/服务器架构下的软件技术的发展现状。

第二章：网络爬虫技术。对网络爬虫技术进行系统的阐述，网络爬虫如何模拟用户行为，介绍网络爬虫的几种搜索策略，并对多个开源爬虫框架进行介绍。

第三章：基于客户端/服务器模式下的软件开发，阐述什么是 C/S 架构，解释其通信流程，与使用的底层网络协议，以及如何使用 IO 复用提升服务器性能。

第四章：基于 python 网络爬虫的天气搜索与预测工具的设计与实现。主要介绍天气搜索工具的设计步骤与实现细节，并对爬取结果进行处理与分析。

第二章　网络爬虫技术

2.1 网络爬虫的基本架构及工作流程

网络爬虫（Web crawler）是一种能够自动提取网页信息程序的统称。爬虫程序一般分为数据采集、数据处理、数据存储三个模块。网络爬虫的定义有狭义与广义之分，广义爬虫的定义为：所有能够利用 http 协议搜索 web 文档的软件都称之为网络爬虫，狭义爬虫的定义为：利用标准的 http 协议，根据超级链接和 Web 文档检索的方法遍历万维网信息空间的软件程序[1]。

网络爬虫的功能是根据已有 URL 去爬取对应的网站，并从得到的 html 文档中提取出用户需要的信息和新的 URL,然后就可以进行递归的爬取。所以说，爬虫就是一个模拟用户使用浏览器发出 http 请求的程序。举个例子，虽然 Scrapy 爬虫与 Chrome、Firefox 等浏览器是两种不同的客户端程序，但它们都用一样的方式请求与下载网页：客户端程序先连接到 DNS 域名解析服务器上，域名解析服务器将这些客户端程序传给它的域名翻译成 web 服务器的 IP 地址。HTTP 协议默认的端口是 80 端口，是一个知名端口。然后客户端通过 IP 地址和端口去向 web 服务器发起 http 请求，http 协议是 OSI 七层模型中的应用层协议，它使用的底层协议分别是 TCP（传输层协议）和 IP（网络层协议）。连接被发起之后，客户端的请求被对端的 web 服务器监听到，它进行处理并将处理结果返回给 Client 程序[8]。

但是浏览器和爬虫程序还有着本质的区别的，因为浏览器需要用户的点击来完成操作 http 请求，而爬虫程序是自动完成 http 请求的，所以爬虫程序需要有一整套完善的架构才能完成全部的工作任务。

爬虫程序的结构框架，基本就是由控制器，解析器，资源库、中间件四部分组成的。图 2.1 是爬虫的工作流程图与步骤解析。

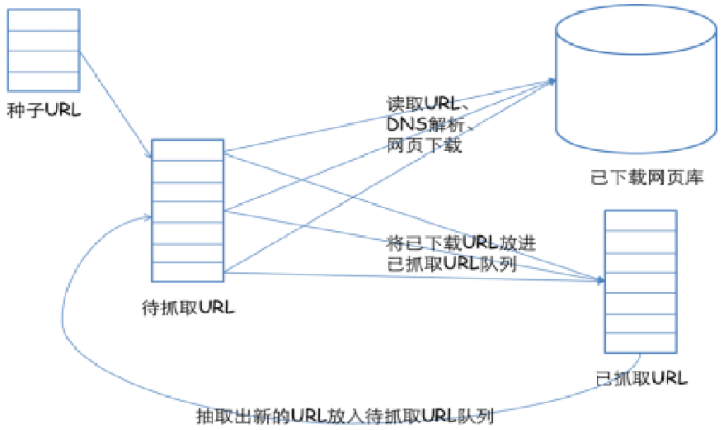

图 2.1 爬虫的工作流程

爬虫技术已经发展了十几年，整体框架上已经相对成熟，下面是通用的爬虫框架工作流程：

1）首先从所要爬取的网站中挑选一些合适的网页，将这些网页的 url 作为初始 url（也成为种子 url）

2）将初始 url 放到一个待抓取 url 的队列中，之后调度器将从这个队列中取出 url 进行网页下载。

3）爬虫程序从待抓取 url 队列中一次读取 url，并将 url 通过 DNS 解析转换为 web 服务器 IP 地址

4）再由网页下载器根据 IP 地址和 url 中域名后的相对路径，发起 http 请求，下载目标网页

5）已经下载的网页，爬虫程序会将这个网页的 url 信息存入到一个已爬取 url 库中，这个已爬取 url 库记录着所有已经被 download 的网页的连接,通过这种机制，可以防止重复多次抓取同一个网页。

6）爬虫程序会对下载好的 html 文档进行分析，将其中的 url 链接提取出来，使用 Bloom Fliter 之类比较高效的算法，测试已爬取队列中是否存在这个 url，如果不在，则说明这个一个未被爬取的 url，那么就将它放入带爬取队列中。

7）调度器再次从带爬取队列中取出 url 进行爬取，如此形成一个完整的循环，直到待爬取队列为空结束。

2.2 网络爬虫的搜索策略

网络爬虫从一些有价值的链接地址出发，首先获取初始网页上的 URL，在抓取网页的过程中，通过正则表达式或 xpath 技术，提取出网页内的 url 地址，并将新的 url 地址放入带爬取队列里，重复这个过程直到爬取完全部页面，或满足预设条件停止爬取[2]。然而由于网络资源太过于庞大复杂，无数网页资源之间的关系盘根错节，就算是再精明的网络爬虫，可能也会在这网络森林中迷失方向，所以我们需要给爬虫从网页上爬取数据的过程设置一些搜索策略，以便更好的进行数据搜集工作。

网络爬虫通过请求 html 文档访问某一站点，遍历 web 空间，不断从一个站点跳到另一个站点，通过搜索与分析网站结构，将网页内的信息下载、归类、建立索引，再将索引关系加入到数据库中[3]。通过使用 HTML 语法结构来定位与搜寻目标信息和指向其他网页的链接地址。通过这种方式，爬虫程序能够不依赖用户的干涉实现自动爬取[11]。

2.2.1 深度优先搜索策略

深度优先搜索策略（DFS）的原理就先向深层第递归搜索，从起点开始尽可

能深的搜索图。在计算机科学的图论中，深度优先就是对于一个节点来说，如果这个节点还有以它为起点而且尚未被探索完的边，那么就继续从这个节点探索下去，直到以这个节点为起点的所有边都已经被探索过，搜索将返回到这个节点的上一层。这个过程会重复执行直到所有从 root 节点开始的边都被探索完毕。但是深度优先搜索策略可能会使爬虫递归搜索一个网站的层次太过于深入，并不利于搜索离门户页面较近的网页信息，而且有的时候可能会导致爬虫陷入过深的递归搜索中导致爬虫出不来的情况。

2.2.2 广度优先搜索策略

广度优先搜索算法（BFS）是原理最简单明了的搜索算法之一，其核心思想就与深度优先搜索相对，也就是优先横向遍历一个节点的所有孩子节点，每访问一个节点，都将这个节点的所有孩子节点都加入一个队列中，按照“层次”遍历整颗节点“树”。同时 BFS 这种算法也是很多重要的图论算法例如的原型，例如 Prime 最小生成树算法与 Dijkstra 有向图的单源最短路径算法，都采用了类似广度优先搜索类似的思想[4]。

2.2.3 聚焦搜索策略

聚焦搜索策略的出现，主要是因为随着互联网的不断发展与不断扩大，网页信息开始呈指级增长，那么传统的搜索策略在面对如此海量与庞大的网页信息会显得力不从心，因为其中掺杂了大量的与想要获取的信息关联性较低的或毫无关联内容，面对这种情况，就产生了能够定向抓取网络资源的聚焦搜索爬虫技术。聚焦搜索策略只会搜索与目标主题相关的网页。聚焦爬虫会根据某种算法给它所下载下来的网页进行评分，然后进行排序，排序越靠前的，会越先被访问[10]。这种搜索策略会保证爬虫程序能优先追踪最有可能含有目标页面链接的页面，聚焦搜索策略的关键是怎样合理有效的评价一个 url 的价值。

一个网页链接的地址存在于另一个网页页面上，所以一般来说价值较高的网页页面内存在的网页链接接地址的价值也是比较高的，例如像 hao123 这样的导航网站，就包含了很多能够有很高评分的门户网站的地址。但有的时候，聚焦搜索策略也可能会因漏掉一些评价不高，但是可能确实包含了有用信息的相关网页，所以不能盲目使用聚焦搜索策略，还应针对问题具体分析，才能得到最好的抓取效果。

2.3 网络爬虫伪装用户

随着互联网的发展，网络爬虫逐渐普及。如果一个网站的内容比较有价值，那么它会被网络爬虫爬取的几率是很高的。比较智能的搜索引擎的爬虫的爬取频率是比较低的，这种爬虫在爬取网页的时候消耗的网站资源比较低。但仍旧有一些肆无忌惮的爬虫程序，开启大量线程并发的抓取网页，给网站带来很大负担，

让普通用户的使用体验受到影响，甚至于无法访问网页。

网站反爬虫的策略一般有以下几种：报文头部（Headers），用户行为（IP 访问平率，进行相同操作的频率，cookies），以及网站的数据的加载方式。

检查报文头部（Headers）比较常见的反爬虫方法。用户行为检测是有些网站是通过检测用户在网页上所做的操作是否合理来判断的。假如一个 IP 地址短时间内尝试多次访问一个网页，假如一个账号短时间内多次进行了相同的操作，那么很有可能会被认为是网络爬虫而被采取相应措施。很多网站必须用户登录后，才会将信息通过 html 文档传递给浏览器，若没有登陆的话，是得不到有用的信息的。例如新浪微博，不登录账号，只能看到大 V 的前十条微博，想要看到全部内容，网站要求用户必须处于登录态，也即需要 Cookie 验证，才能获取到相关数据。

那么，当我们需要爬取这个网站上的数据的时候，这些反爬虫策略我们要找办法绕过去。我们可以把爬虫程序的爬取过程尽可能的模仿成浏览器的正常操作，也就是贴近一名真实用户的上网行为，就可以将被网站的反爬策略发现的概率降到最低。通过以下几种策略，可以将爬虫进行伪装。

2.3.1 User Agent

分析 http 请求的报文头部的信息，是最常见的反爬虫策略。大部分站点都对 User-Agent 进行检测，看是否存在 User-Agent 或者 User-Agent 是否合法，正常用户使用浏览器上网，所产生的报文头部中一定是有 User-Agent 部分的。如果网站通过检查 User-Agent 作为爬虫判断策略，那我们可以在爬虫的代码中添加 Headers 模块，并将一个合理的 User-Agent 到爬虫的报文头部中。

2.3.2 IP 代理

有些网站会针对用户行为进行反爬虫。例如使用相同的 IP 地址，如果短时间内多次尝试访问同一域名下的页面，或者同一个账号短时间内多次进行了类似操作，若发生这些情况，那么很有可能会被认为是网络爬虫。对通过检测 IP 频率的情况来说，可以使用代理 IP 工具对软件进行代理，或我们自己抓取一部分代理 IP 并使用 requests 和 urllib2 库来解决。针对这种情况，我们可以专门维护一个代理 IP 池，有了可替换的代理 ip 之后，我们可以每对目标站点进行几次爬取，就更换一个 ip 地址，这样可以很简单的绕过检测 IP 频率的反爬虫策略。对于限制同一账号连续相同操作次数过多的情况，可以在每次发出请求后，隔一段时间再进行下一次爬取。也可以发出几次请求之后，进行登出与登入操作，然后继续运行爬虫程序。用这种办法来防止一个账号进行相同请求次数过多容易被反爬虫机制发现。

2.3.3 Cookie

部分站点需要用户登录后才能访问某个页面和获取定制化的信息，否则网站的网页源码中就没有你想要获取的内容。所以，我们可以先在浏览器中进行登录，然后使用开发者工具将 cookie 复制出来，在爬虫发出的请求中，将 Cookie 带上，就可以模拟出用户登录了，从而获得登陆后才可以得到的信息。

2.4 几种开源爬虫框架

分网络爬虫框架主要可以根据规模分为两种：分布式爬虫，和单机爬虫。

Nutch 是一种比较出名的分布式爬虫。但是这种爬虫框架并不适合普通用户的需求，Nutch 的适用范围主要是搜索引擎，Nutch 差不多有三分之二的工作流程是为了设计一款搜索引擎，它对精确地数据搜集没有太大的意义。如果你想要修改 Nutch，来让它很好的适用于精确数据搜集的业务，那可能会是一项比较大的工程，会破坏其原有结构，而且会把它改的面目全非。有修改 Nutch 的能力还不如重新编写一个符合自己需求的分布式爬虫框架。Nutch 是基于分布式计算运行的，如果服务器数量较少，那就发挥不出来分布式计算的能力，这种情况下的分布式爬取速度可能还比不上单机爬虫。

单机爬虫，单机爬虫的开源框架有很多种，比如使用 C++语言编写的 Larbin 框架、使用 python 语言编写的 Scrapy 框架等。Larbin 可以追踪页面上的 url 并抓取，它是功能比较单一的爬虫框架，只提供了抓取网页的基本功能，解析数据、存储数据的事情需要用户自己完成。所以我们可以针对自己的实际需求对 Larbin 进行二次开发，这种定制化的爬虫肯定是非常高效的。再来说说 Scrapy，Scrapy 是一个基于 Python 的爬虫应用框架。 Scrapy 可以被应用在大数据处理、数据挖掘等应用中。 这个框架的最初设计目的是页面抓取， 也可以用在获取 API 所返回的数据或者编写通用的爬虫程序之中。图 2.2 是 Scrapy 的结构图，以及 scrapy 的各个组件在系统中发生的数据流向的概览。并对每个组件都做了一些简单的介绍。

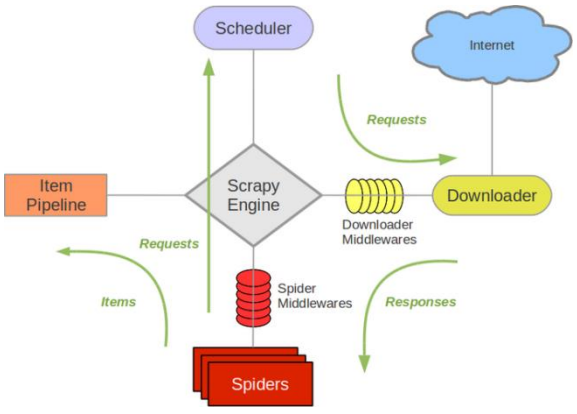

图 2.2scrapy 框架结构

Scrapy Engine 引擎是用来控制整个 Scrapy 系统的数据处理流程的核心部件，也会进行事务处理的触发。Scheduler（调度器），调度程序从 Scrapy 引擎接受请求并将请求进行排序，再列入队列，并在 Scrapy 引擎发出请求后，将结果返还给他们[5]。Downloader 的根据 url 地址抓取网页，并将网页内容传给 Spiders。Spiders 蜘蛛由用户自己定义与实现，其中的 parse()函数会使用 xpath 或正则表达式技术对 html 文档的内容进行解析，并抓取指定位置的 url，对抓取到的 url，会继续传给下载器下载，然后被其他的处理函数进行解析。每个 Spiders 都可以处理一个域名或一组域名。可以给每个 Spiders 都定义一组特定网站的解析规则和抓取规则。

2.5 网络爬虫的几种应用

行随着互联网的蓬勃发展，每天都有海量的数据汇入其中，但是这些数据是分布在无数个网页上的，用户如果有需要查找的信息，那么就需要有一种能有效而快速的信息搜寻方式为其提供服务，有时候人们需要快速、便捷的获取和一些关键词相关的信息，这时候搜索引擎就能够为我们提供帮助，例如百度搜索、谷歌搜索等，这些搜索引擎也是通过无数个爬虫，将网络上林林总总的信息全部进行了爬取，用户如果需要知道天气，那么百度一下天气关键词，就可以立刻得到各种天气网站的链接地址。除此之外，如果用户有一些个性化的数据采集需求，比如我想要知道知乎网站上，点赞数排名前十的问题与回答，或者我想要对新浪微博上一段时间之内的所有微博做一下数据分析，那么传统的使用关键词搜索的方式是无法满足这种需求的，这种情况下，就需要进行个性化订制的爬虫，才能够满足形形色色的需求，毕竟互联网包含了无数、海量的数据，通过网络爬虫进行合理的挖掘与分析，可以让其隐藏的价值得到更充分的体现。

第三章　基于客户端/服务器模式下的软件开发

3.1 客户端/服务器模式介绍

客户端／服务器(Client/Server，也即 c/s)模式，是基于网络资源的不对等，为了实现资源共享而提出的一种模式。客户端服务器模式有比较强的事务处理能力，一些任务会被放到客户端上计算之后交给服务器，从而大大降低了服务器的运算压力。

图 3.1 是 c/s 模式的比较简单的示意图，由两部分构成：客户机通常是用户的个人电脑端。服务器则处理客户端发送的请求。

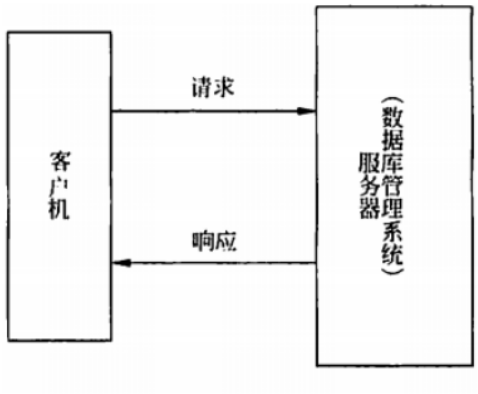

图 3.1　客户机/服务器模式

这种模式把客户端与服务器区分开来。主动的发送出请求的是客户端，而被动的进行响应的一方是服务器。有的时候，服务器可能不止一台，一台服务器无法完成某项工作，那它就会向其它服务器发送请求，这种情况下，发送请求的服务器，就相当于是另一台服务器的客户端。从一个连接建立的角度上看，主动启动通信的是客户端，被动的等待连接的是服务器。

3.2 客户端与服务器的通信过程及原理

要了解客户端与服务器的通信方式，首先要了解一下 Socket。Socket 被翻译为“套接字”。举个例子，网络通信像打电话，首先要有一部电话机，而且还得知道对方的电话号码。通话双方是两个正在运行的程序，电话就相当于套接字，而电话号码就是对方的 IP 地址。用拨号盘呼叫对方，就相当于是作为客户端主动发起了连接请求，如果另一端此时处于空闲状态，电话响了那他就会拿起话筒，这就相当于连接建立成功。而双方通话的过程就相当于对 socket 读写数据。

既然这个通信的流程是双方一起建立的，那么一定要有一套规范的建立起连接的过程。下面就是客户端与服务端通过 Socket 建立起 TCP 连接和关闭连接的过程。

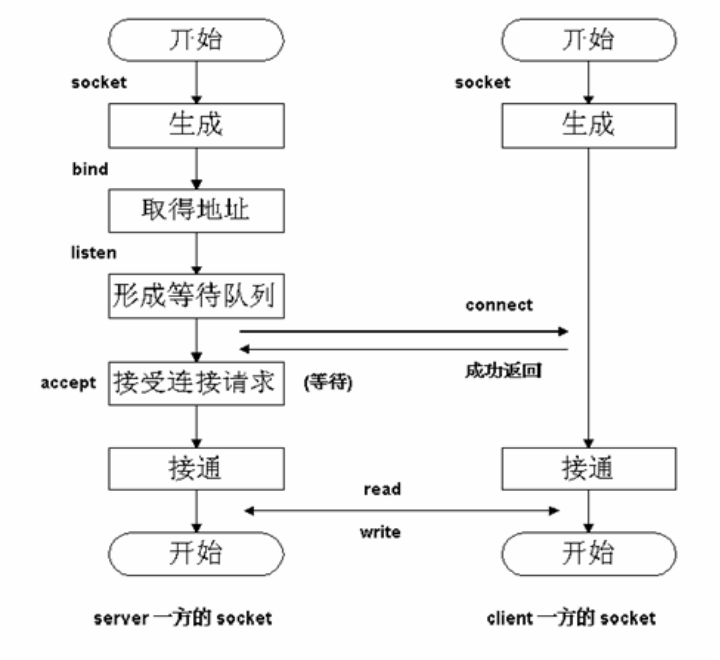

图 3.2 socket 建立连接的过程

如图 3.2，服务器端调用 socket 创建一个套接字，而且这个套接字是用来监听自身的端口的，所以需要用 bind 函数将 ip 和 port 告诉它，之后这个 port 会被服务器监听，因为服务器不知道谁会连接自己，也就是说服务器不知道下一个连接自己的客户端的信息，所以它只能事先和客户端约定达成一致性：客户端会将请求发送到服务器指定的某个端口，那么当请求到达后，服务器就可以监听到这个请求了。完成一些初始化工作后，服务器会调用 accept，accept 是一个阻塞函数，它会一直阻塞直到有客户端发出的请求到达被监听到的端口。

下图 3.3 为 TCP 三次握手的详情：

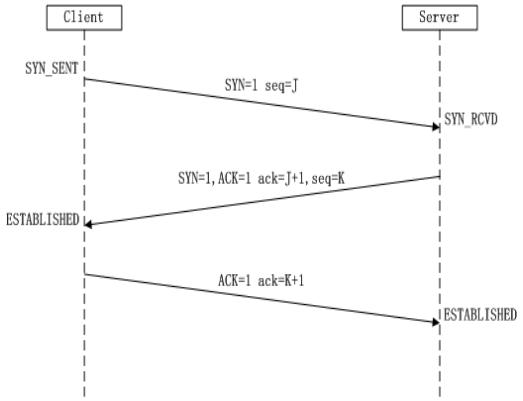

图 3.3 TCP 三次握手

有请求到来时，服务器从 accept 函数的阻塞状态返回，调用 read 函数，读取客户端通过 TCP 协议传输过来的内容，因为 TCP 协议是可靠地协议，所以一般来说无需担心内容会出错。这个被调用的函数 read 是个同步阻塞函数，会阻

塞直到对端有数据传来。当然，也可以使用同步非阻塞版本的 read 函数，但前提是要放在一个 for 循环中判断数据是否到来。

客户端将数据发送完毕，会调用 close 函数单向关闭连接，对端协议层若接收这个关闭信号(FIN)，那服务器会通过 read 函数的返回值为 0 来判断客户端关闭了连接，既然客户端已经关闭了连接，那服务器也会调用 close 关闭连接，如图 3.4。

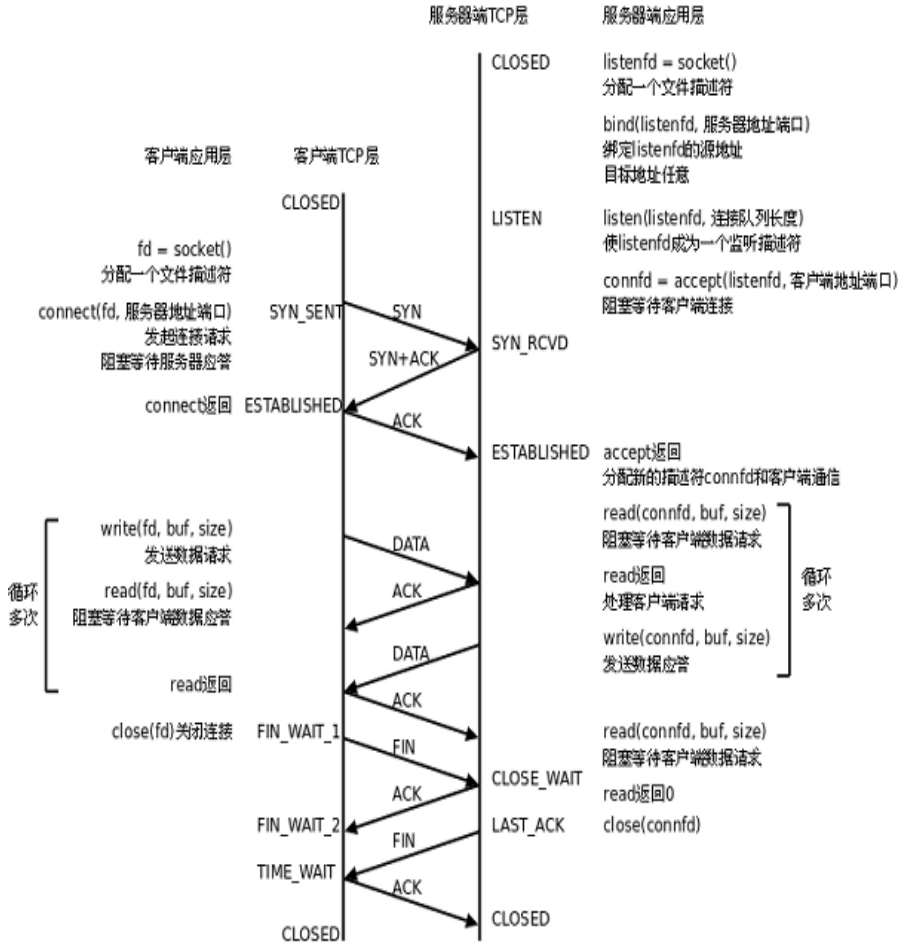

图 3.4　TCP 连接从开始到结束的全过程

需要注意，服务器端正常情况下有一个监听套接字和至少一个已连接套接字。监听套接字是比较特殊的一个套接字，用来接受新的连接请求，并会返回一个已连接套接字。已连接套接字被用来和客户端进行通信，一个已连接套接字就意味着一条 tcp 连接。双方通过向套接字写数据和从套接字读数据来实现两个网络进程之间的通信。

3.2.1 底层网络协议

网络编程中使用 Socket 进行多机之间的互联与通信，linux C 下，使用 socket 函数创建套接字的时候，可以选择要是用的网络协议。Socket 可以支持不同的

OSI 第 4 层（传输层）协议，包括大名鼎鼎的 TCP 传输控制协议和 UDP 用户数据报协议。TCP 是面向连接的协议，每条被发送的消息都需要对端进行确认，才能证明这条消息对方接收成功，所以可靠性较高。而 UDP 是面向无连接的协议，也就是说，它在发送消息之前，不需要建立连接，可以直接将消息发送到目标地址，因为对端不需要确认消息，因此 UDP 协议不是可靠的协议，相比来说，使用 UDP 传输数据，速度要快一点。但是，出于对所传输的信息的可靠性的保证，我们选择了 TCP 协议来进行本课题的底层网络协议。

图 3.5 是 TCP 报文格式的内部结构：

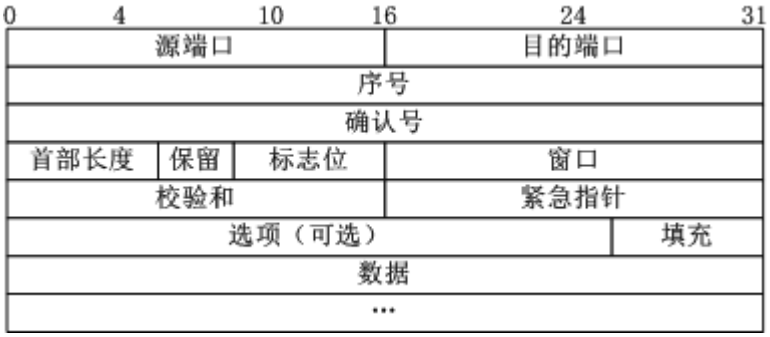

图 3.5　TCP 报文的结构

TCP 协议“三次握手”的目的，是根据端口和 IP 地址在 client 和 server 之间建立起一条 TCP 连接,同时会做一些信息交换，比如 TCP 滑动窗口大小。因为 TCP 有打开拥塞控制的机制，就是通过这个值来进行流量控制的，所以滑动窗口的大小，对通信双方来说，都是一个一直在动态调整的数值，需要及时告诉对端[9]。在网络编程中，TCP 传输控制协议的三次握手发生在客户端调用 connect() 函数的时候。

3.2.2 I/O 多路复用

在 linux 下编写服务器程序的时候，我们不得不考虑的一个问题就是，服务器程序是否可以同时处理多个客户端的发起的连接请求。

单线程的进程的执行过程是线性的（不考虑使用协程），那么它在调用一些系统 I/O 函数：write、read 的时候，线程会陷入系统调用出不来，阻塞在内核中，无法继续向下执行别的操作。若只是一对一的 client/server 之间的回合式通信（就是交叉发送消息），这是可以运行的，但是在一个服务器面对多个客户端的连接的时候，若服务器被阻塞在一个对客户端的 I/O 操作中，那么当另一个客户端的连接请求到达的时候，被阻塞的服务器程序是无法处理的。所以问题就在这里，单线程的服务器无法处理多个客户端的连接请求。如果是同步阻塞模型下使用多线程的方式的处理，那就是在主线程中创建一个循环，每当有新的客户端的请求到达的时候，主线程监听到这个请求，会创建一个新的线程，并将这个与这个会

话相关的套接字作为参数传给新的线程，从而让新的线程全权负责处理这个客户端的请求，新的 IO 请求会阻塞在新的线程中，但这种同步阻塞模型下的多线程的方法最致命的一个缺点是，每个进程虽然在理论上拥有全部内存大小的虚拟内存空间，但是实际上操作系统也是要占用一部分内存的，而且每一个线程都拥有自己的线程栈，这个栈的大小一般来说是固定的，这就意味着，一个进程能开辟的线程数量是有限制的，一般来说，4G 大小的 linuxx 系统的计算机，能最多开辟出不到两千个线程，这就意味着，如果采用这种方法，服务器最多只能为不到两千个客户端服务，是不具有上万的并发能力的。遇到这种情况的时候，我们就必须使用 I/O 多路复用技术来解决问题。

I/O 多路复用会调用内核提供的函数 select()。在传统的同步非阻塞模型中，我们通过使用循环加判断可读的方式，防止进程阻塞在内核 I/O 操作里。事实上，select 也会使进程阻塞。图 3.6 是 select 说明图。

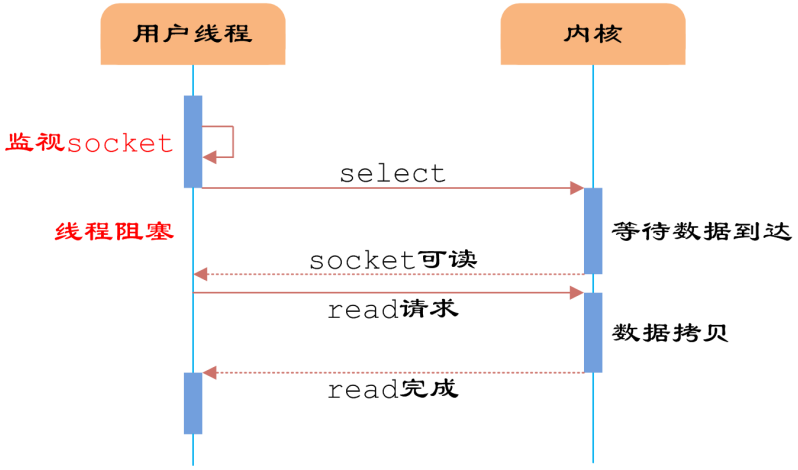

图 3.6　I/O 多路复用：select

Select 被调用后会先陷入内核态，当有 socket 处于活动状态的时候才会返回用户态，select 可以同时检测多个读写的 I/O 操作，三个文件描述服集合分别是 read_set，write_set，exception_set，内核监视所有三个集合中的文件描述符的状态。只要有一个 socket 的数据准备好了，select 就会返回套接字可读的消息，从内核态返回到用户态，接着便可以调用 recvfrom()函数读取数据了。正因为 I/O 多路复用模型可以同时阻塞多个 I/O 操作，可以同时键测多个描述符的状态，而这是在同步阻塞 IO 模型中必须使用多线程的方式才能够达到的效果，所以才把这种技术成为 I/O 多路复用。

第四章　基于网络爬虫的天气搜索工具的设计与实现

4.1 总体框架介绍

况本课题的软件架构设计，主要包括以下几部分。

服务器：包含一个用来保存天气信息的 Mysql 数据库，和一个长期运行的用来相应客户端请求的 Server 程序，以及使用 Scrapy 框架编写的网络爬虫程序。

客户端：使用 Qt 编写的客户端，用户登录与查询天气都在这个客户端上操作。

图 4.1 是天气搜索工具软件的各个功能模块：

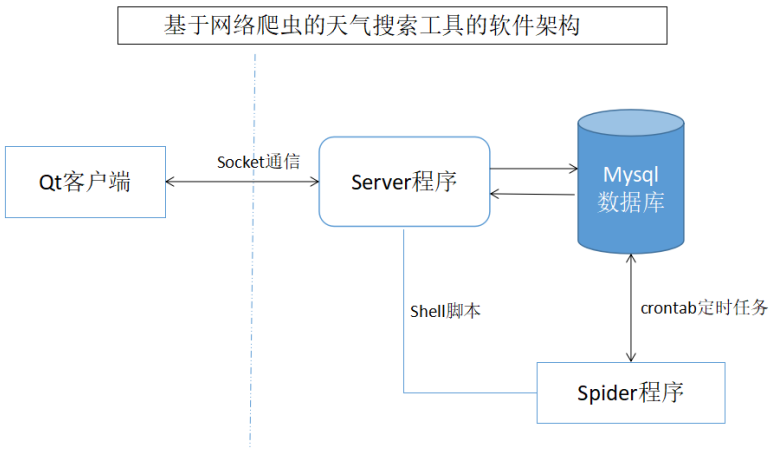

图 4.1　软件架构

4.2 数据爬取

数据爬取部分，是由 Python 编写的的网络爬虫程序实现的。网络爬虫程序在编写好后，可以自动的、不受用户干扰的运行，并从目标站点搜索与下载内容。

本课题使用的框架是 Python 的开源爬虫框架 Scrapy，它被用于抓取 Web 节点并且能通过 Xpath 技术从 html 文档中提取结构化的数据与信息。它最大的好处是提供了很多类型的爬虫基类，我们可以根据个人的需求进行选择和修改。

爬虫代码的文件结构组成如图 4.2：

```
root@by-virtual-machine:/home/by/python_test/today_weather# tree ./
./
├── cron.sh
├── scrapy.cfg
├── show
├── today_weather
│   ├── __init__.py
│   ├── __init__.pyc
│   ├── items.py
│   ├── items.pyc
│   ├── middlewares.py
│   ├── pipelines.py
│   ├── pipelines.pyc
│   ├── settings.py
│   ├── settings.pyc
│   └── spiders
│       ├── __init__.py
│       ├── __init__.pyc
│       ├── weatherSpider.py
│       └── weatherSpider.pyc
└── weather.log

2 directories, 17 files
```

图 4.2　爬虫项目代码结构

对爬虫项目的主要文件的说明如下：

scrapy.cfg：这个文件是整个项目的配置文件。主要是给命令行操作提供了一些配置参数。

items.py：这个文件包含了设置数据如何存储的类，让数据结构化，如天气、风速、日期等数据。

pipelines.py：用于处理从爬虫 Spider 返回的数据。

settings.py：这个文件是整个爬虫项目的配置文件，包含了很多配置内容，比如延迟下载的时间，是否遵循网站的 robots.txt 的口令约束，以及并发数量等。

spiders：这是爬虫程序所在的目录，里面有爬虫的主体文件。

middlewares.py:下载中间件是处于引擎和下载器之间的一层组件，可以有多个下载中间件同时被加载运行。

cron.sh:自己编写的定时脚本任务，用来定期进行数据库内容的保存，以及爬虫运行日志的保存。

weather.log:保存爬虫程序的运行日志，以便出错误时方便定位错误。

爬虫项目的总体框架基本就是这样，接下来将介绍各个模块的功能是如何用 Python 语言来实现的。

4.2.1 网站结构的分析

本课题中，爬虫所爬取的网站是中国天气网 http://www.weather.com.cn，爬虫程序从天气网站中爬取相关的城市的天气数据，而在设计爬虫程序之前，我们

需要先对这个站点的网页结构进行分析，了解各个部分的布局，以及关键的天气信息在网页中的位置。

分析网页需要知道 HTML 的具体结构，HTML 的全称是“超文本标记语言”[7]，所谓的标记就是通过很多个标签来描述一个网页的具体部分。以<>开始以</>结束的标志都是标签。这些标签通常是成对的出现，标签的内部也可以嵌套子标签，也即类似于先辈-子孙的关系。位于最外面的 html 标签是所有标签中级别最高的，它没有父节点。Head 节点、body 节点次之，层层嵌套，最后才是包含了具体内容的文本、链接等信息。一些常用的标签如下：

<head>：这个节点包含了很多与网页相关的重要信息。例如，网页的标题在<head>下的<title>中定义，我们也可以在<head>下的<style>节点中定义对网页内容使用的 CSS 样式。

<body>：这个节点包含了更多的子节点，基本上我们都可以看到的网页内容（文本、链接、图片等）都在<body>节点中。例如：

<P>标签，表示一个段落；

<H1>标签，它的内容是文本部分的标题；

<a>标签，内部包含了 url 链接；

<Img>标签，表示包含有图片链接；

<Form>标签，表示一个表单；

标签，表示一个区块。

知道网页结构与标签的含义，我们已经可以对中国天气网的网页源码进行具体分析了，从而写出爬虫去抓取这些内容。

图 4.3 是中国天气网的部分网页源码：

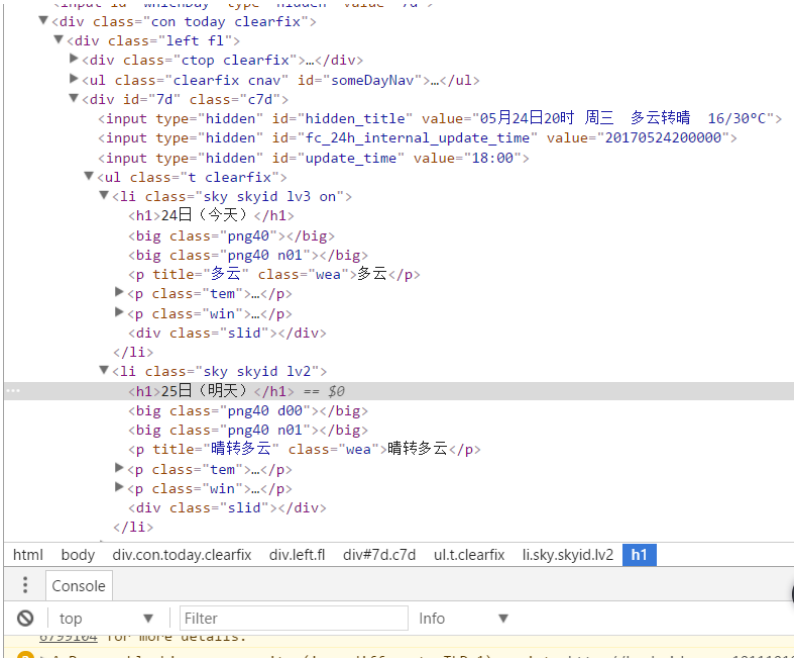

图 4.3　网站 html 文档源码

从中可以看出，西安近七天的天气信息，是保存在 标签下的<ul class="t clearfix">标签的第一至第七个<li class="*">子节点中的，图4.4 用一个具体的例子来进行分析某一天的天气数据：

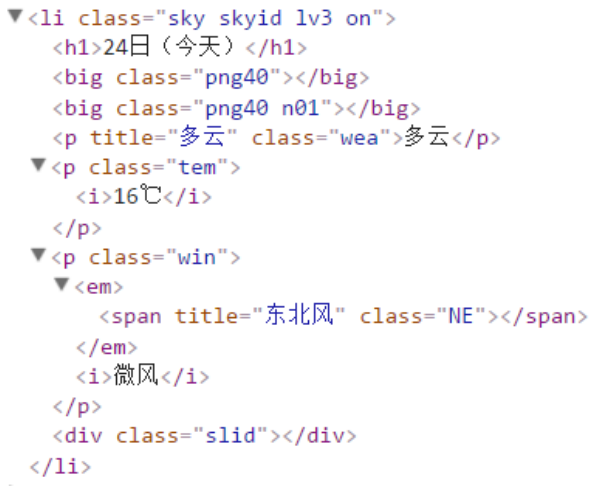

图 4.4　数据位置分析

从图中可以分析出，<li>节点嵌套了一些子节点，这些子节点包含了 24 日这一天的全部天气数据：

<h1>节点，日期

<p class=”wea”>节点，天气气象

<p class=”tem”>下的节点：最高温度；<i>节点：最低温度

<p class=”win”>节点，节点：风向；<i>节点：风速

这是对网页上“七天”按钮所链接的网页内容的分析，通过分析，现在我们已经知道 标签下的节点都包含得的信息以及信息位于什么位置。

要获取这些数据，我们可以通过 xpath 技术进行节点的定位：

```
sel = response.xpath('//*[@id="7d"]/ul/li')
item['weatherWea']=sel.xpath('p[1]/text()').extract()
item['weatherTem1']=sel.xpath('p[@class="tem"]/span/text()').extract()
item['weatherTem2']=sel.xpath('p[@class="tem"]/i/text()').extract()
item['weatherWin']=sel.xpath('p[@class="win"]/i/text()').extract()
```

通过这种办法，成功的提取出了未来七天的天气气象、风向、风速等信息，并将其根据 items.py 中定义的字典结构，添加到 item 内，传给 pipeline 进行解析与存储。

但是只知道近七天的大致天气走向是不够的，我们还需要获得每一天的各个

时段的天气走向，需要获得实时的天气数据，所以接下来还需要对“今天”按钮所链接的网页的源码进行分析。

通过 Chrome 的元素审查，找到了一天内各个时段的天气走向的标签的位置：

图 4.5　各小时温度预测

从图 4.5 可以看到，所有内容位于 div<class=“curve_livezs”>节点下，其中温度位于 div<class = tem>的子节点，时间位于 div<class=”time”>下的子节点。如果我们能够得到这两部分内容，就可以得到某一天之内各个时段的详细的温度走势了。

所以，我再次根据通过这种分析方法，得到了信息的 xpath：

item['weatherTime']=response.xpath('//*[@id="curve"]/div[1]/em[%d]/text()' % i).extract()

item['weatherTem']=response.xpath('//*[@id="curve"]/div[4]/em[%d]/text()' % i).extract()

item['weatherWea']=response.xpath('//*[@id="curve"]/div[6]/em[%d]/text()' % i).extract()

但是后来运行爬虫程序的时候，发现无法通过这些节点的 xpath 获取到任何数据，所以进行了深入的探究，我发现网页的源码中并没有这些节点，但是在 Chrome 的元素审查中却有这些节点，并且可以通过开发者工具提取出这些节点的 xpath。于是我意识到，这部分的节点实际上是由浏览器解析网页源码并动态加载出来的，不是实际存在于网页源码中的内容，而爬虫程序只会下载网页源码并对网页源码进行解析，没有做浏览器动态加载 JS 生成新的内容那一部分工作，

所以爬虫程序是无法获取到这部分信息的。

但是，这部分信息虽然是浏览器动态解析加载出来的，可原始的数据一定是存在于源码中的，所以我再次对网页 html 代码进行分析，发现全部的天气数据都在一个标签内，这个标签的位置图 4.6：

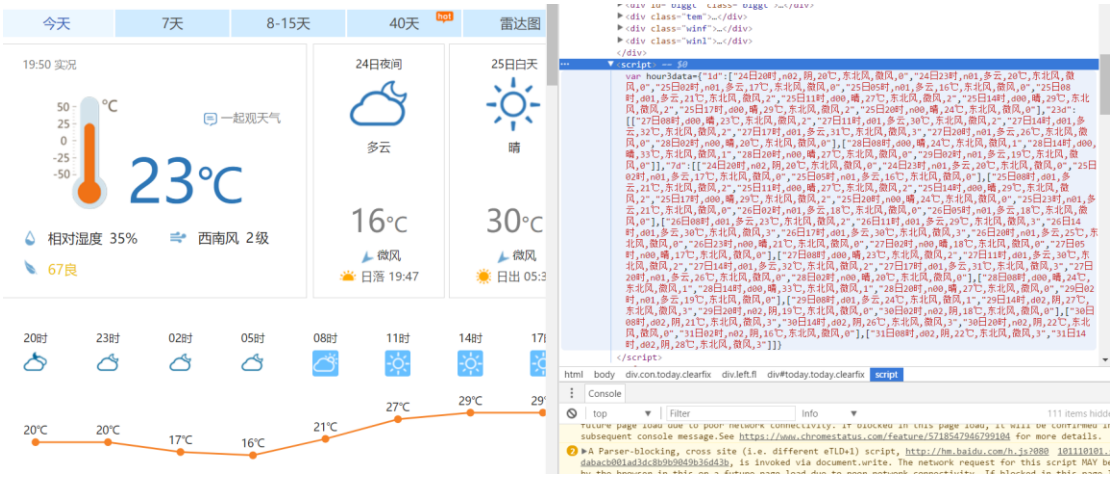

图 4.6　网站结构分析

从而可以提取出它的 xpath：

sel.xpath('//*[@id="today"]/script/text()').extract()

所以，未来七天的天气走向，和每日的各时段的天气信息都可以通过 xpath 分析得到其内容，下一步就可以进行爬虫程序的设计与实现了。

4.2.2 Spider 的设计实现

网络爬虫的各个模块的功能在 4.2 节已经做了简单的介绍，下面详细描述其实现细节。

Items.py 主要包含了一个类 TodayWeatherItem，其功能是配置需采集页面的字段实例。

```
class TodayWeatherItem(scrapy.Item):
    content     = scrapy.Field()
    city_code = scrapy.Field()
```

因为中国天气网上一个页面只有一个城市的详细天气信息，而且上一小节提到了，网页上“今天”按钮所链接的网站的一些标签都是浏览器动态加载生成的，而这些标签的内容全部都保存在一个<script>标签中，所以爬虫只需要每次从网页中获两个标签内容就可以，一个是<script>内的 json 串 content，另一个是从当前网页源码中得到这些天气数据所属的城市，也就是获得城市天气代码 city_code。

settting.py 主要包含的是一些配置项，以及一些数据库的连接参数。

```
BOT_NAME = 'today_weather'
SPIDER_MODULES = ['today_weather.spiders']
NEWSPIDER_MODULE = 'today_weather.spiders'
DEBUG = True
if DEBUG:
    dbuser = 'root'
    dbpass = '123456'
    dbname = 'bs_db'
    dbhost = '127.0.0.1'
    dbport = '3306'
else:
    dbuser = 'root'
    dbpass = '*****'
    dbname = 'bs_db'
    dbhost = '********'
    dbport = '3306'
#遵循 robots.txt 中的口令约束
ROBOTSTXT_OBEY = True
ITEM_PIPELINES = {
    #'today_weather.pipelines.TodayWeatherPipeline': 300,
    'today_weather.pipelines.MySQLStorePipeline': 400,
}
```

weatherSpider.py,是爬虫程序的主体，用来爬取网页内容，并存入 item 中。

在设计上，爬虫程序需要实现这样的功能：从数据库读取用户注册的城市信息，再根据城市天气代码生成 url，将 url 传给爬虫进行网页下载，再对下载在的内容使用 xpath 技术提取出我们想要的内容。所以，这个爬虫程序是分为好几各模块的，摸一个模块都有各自的功能，我会分开介绍。

get_urls()函数，用来从数据库中读取用户注册的城市列表，从而根据城市天气代码生成目标 url，再根据 url 进行网页爬取。图 4.7 是 get_urls()函数的实现。

```
import MySQLdb
from ..settings import dbuser,dbpass,dbname,dbhost,dbport
#从 settings 中，扩展出几个数据库连接参数，供爬虫程使用。
```

```
def get_urls():
    try:
        conn=MySQLdb.connect(user=dbuser, passwd=dbpass, db=dbname,
                             host=dbhost, charset="utf8", use_unicode=True)
        cur=conn.cursor()
        cur.execute('select * from register_city;')
        s=[]
        results = cur.fetchall()
        result=list(results)
        for r in result:
            #将数据库中的城市代码列表，构建成url列表的形式，并返回url列表
            s.append(("http://www.weather.com.cn/weather1d/%s.shtml" % r))
        conn.close()
        return s
    except MySQLdb.Error,e:
         print "Mysql Error %d: %s" % (e.args[0], e.args[1])
```

图 4.7 get_urls()函数的实现

爬虫主体类是 CatchWeatherSpider 类，继承自 scrapy.Spider 类，功能是下载网页并进行内容解析，图 4.8 是其实现。

```
class CatchWeatherSpider(scrapy.Spider):
    name = 'weather' #爬虫程序的名字
    allowed_domains = ['weather.com.cn']
    start_urls = get_urls() #获取初始url列表，将爬取这个列表

    def parse(self, response):
        sel=Selector(response)

        #调试
        sites=sel.xpath('//*[@id="today"]/script/text()').extract()
        print('test--1:%s' % sites)

        #从网页源码中获取城市代码，用来分辨天气记录是属于哪一个城市的
        city_code_path = sel.xpath('//*[@id="someDayNav"]/li[1]/a/@href').extract()
        city_code_path = city_code_path[0].encode('utf-8')
        #city_code_path 的形式是："/weather1d/101110102.shtml"，需要用正则表达式提取city_code
        city_code = re.findall(r'/weather1d/(.+?).shtml',city_code_path)[0]

        #获取网页script节点中存储的全部内容
        contents  = sel.xpath('//*[@id="today"]/script/text()').extract()
        contents  = contents[0].encode('utf-8')
        content_list = re.findall(r'"(.*?)"', contents)
        i = content_list.index("7d")

        for x in content_list[i+1:]:
            item = TodayWeatherItem()
            item["content"]   = x
            item["city_code"] = city_code
            #这条可以在运行结果上看到每一条记录的实际内容(而不是utf-8码)
            print("%s:%s" % (city_code,x))
            yield item
```

图 4.8 CatchWeatherSpider 函数的实现

Pipelines.py 主要包含了一个 MySQLStorePipeline 类，用来处理 CatchWeatherSpider 类的 parse（）函数传来的 item 对象，并对其中包含的数据进行处理与入库，实现如图 4.9。

```
class MySQLStorePipeline(object):
    def __init__(self):
        self.conn = MySQLdb.connect(user=dbuser, passwd=dbpass, db=dbname,
                                    host=dbhost, charset="utf8", use_unicode=True)
        self.cursor = self.conn.cursor()
        #清空表：
        self.cursor.execute("truncate table weather7day_full;")
        self.conn.commit()
    def process_item(self, item, spider):
        curTime =  datetime.datetime.now()
        try:
            self.cursor.execute("""INSERT INTO weather7day_full (city_code,weatherDate, weatherWea,
                                                    weatherTem, weatherWinf,weatherWinl, updateTime)
                                VALUES (%s, %s, %s, %s, %s, %s, %s)""",
                                (
                                    item['city_code'],
                                    item['content'].split(',')[0],
                                    item['content'].split(',')[2],
                                    item['content'].split(',')[3],
                                    item['content'].split(',')[4],
                                    item['content'].split(',')[5], #五和六
                                    curTime,
                                )
            )
            self.conn.commit()
        except MySQLdb.Error, e:
            print "Error %d: %s" % (e.args[0], e.args[1])
        return item
```

图 4.9　MysqlStorePipeline()函数的实现

上面是获得每一天的详细数据的爬虫的实现，还需要对未来七天天气走向的数据进行爬取。所以还需要从网页源码中提取出当前城市的城市天气代码。

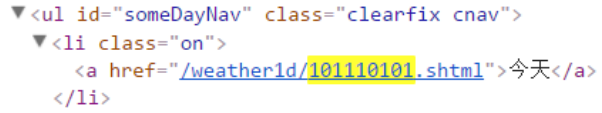

图 4.10　定位城市天气代码

如图 4.10 通过分析网页源码，得到了城市代码的位置，其 xpath 是：

sel.xpath('//*[@id="someDayNav"]/li[1]/a/@href').extract()

再通过正则表达式技术，将连接中的城市代码（高亮部分）取出来：

re.findall(r'/weather1d/(.+?).shtml',city_code_path)[0]

图 4.11 就是爬取未来七天天气走向的网络爬虫的源码：

```
class CatchWeatherSpider(scrapy.Spider):
    name = 'weather'
    allowed_domains = ['weather.com.cn']
    #start_urls = [ "http://www.weather.com.cn/weather/101280101.shtml" ]
    start_urls = get_urls()

    def parse(self, response):
        sel=Selector(response)
        #从网页源码中获取城市代码，用来分辨天气记录是属于哪一个城市的
        city_code_path = sel.xpath('//*[@id="someDayNav"]/li[1]/a/@href').extract()
        city_code_path = city_code_path[0].encode('utf-8')
        #city_code_path 的形式是："/weather1d/101110102.shtml"，需要用正则表达式提取city_code
        city_code = re.findall(r'/weather1d/(.+?).shtml',city_code_path)[0]

        for sel in response.xpath('//*[@id="7d"]/ul/li'):
            item = WeatherItem()
            item["city_code"] = city_code
            item['weatherDate'] = sel.xpath('h1/text()').extract()
            #item['weatherWea'] = sel.xpath('p[@class="wea"]/text()').extract()
            #print(response.xpath('//*[@id="around"]/div[1]/ul/li[1]/a/span/text()').extract())
            #print(response.xpath('//*[@id="curve"]/div[1]/em[1]/text()').extract())
            item['weatherWea'] = sel.xpath('p[1]/text()').extract()
            item['weatherTem1'] = sel.xpath('p[@class="tem"]/span/text()').extract()
            item['weatherTem2'] = sel.xpath('p[@class="tem"]/i/text()').extract()
            item['weatherWin'] = sel.xpath('p[@class="win"]/i/text()').extract()
            yield item
```

图 4.11　获取未来七天温度走向

4.2.3 Spider 的伪装与使用代理 IP

通过使用 User-Agent 的方法，将爬虫程序伪装成普通用户使用浏览器访问，可以一定程度上避免网站屏蔽，在爬虫代码中添加如下部分即可：

head = []

#写入 User Agent 信息

```
head['User-Agent'] = 'Mozilla/5.0 (Windows NT 11.0; WOW64; Trident/7.1; .NET4.0;
                      .NET4.0E; .NET CLR 2.0.41647; .NET CLR 3.1.45934; .NET CLR 3.5.45934;
                       InfoPath.3; rv:12.0) like Gecko'
```

图 4.11　添加 user agent

getProxies_List()是设置代理 IP 的方法，图 4.12 是具体实现：

```
import requests
def getListProxies():
    sessions = requests.session()
    pages = session.get("http://www.xicidaili.com/nn", headers=headers)
    soups = BeautifulSoup(pages.text, 'lxml')#使用lxml解析

    proxyLists = [] #创建列表
    #使用正则表达式匹配技术
    taglists = soup.find_all('tr', attrs={'class': re.compile("(odd)|()")})
    for trtag in taglists:
        tdlists = trtag.find_all('td')
        proxys = {'http': tdlists[1].string + ':' + tdlists[2].string,
                  'https': tdlists[1].string + ':' + tdlists[2].string}
        urls = "http://ip.chinaz.com/getip.aspx";
        try:
            response = session.get(urls, proxies=proxys, timeout=5)
            proxyLists.append(proxys)
            if(len(proxyLists) == 10):
                break;
        except Exception, e:
            continue
    return proxyLists
```

图 4.12　代理 IP

4.2.4 使用多线程提升爬虫性能

Scrapy 框架提供了很多 API 函数与参数设置的选项，scrapy 自带 twisted 线程池，默认是 10 个线程。在配置中可以设置线程数：

CONCURRENT_ITEMS = 100

#在 item Pipeline 中并行处理的最大并发项数。

CONCURRENT_REQUESTS = 32

#Scrapy 下载器最大的并发下载量

4.2.5 Crontab 实现定时数据爬取

载服务器程序是运行在 Linux 操作系统下的，爬虫程序需要定期的运行和更新最新的天气信息，但是由于 scrapy 不自带定时运行的功能，所以需要借助 linux 的 crontab 定时任务来实现这个功能。

图 4.13 是 Crontab 的简介：

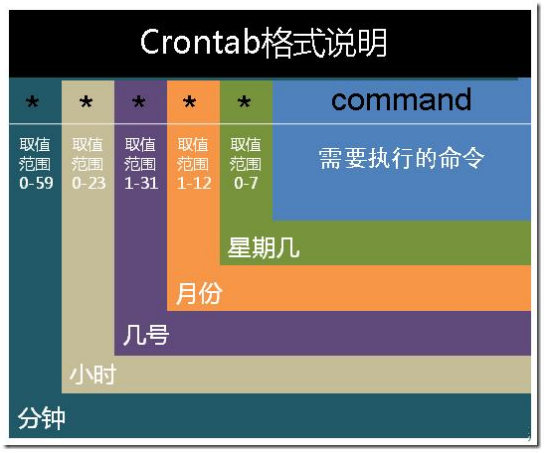

图 4.13　Crontab 定时任务

下面是根据 crontab 语法编写出的 crontab 定时任务：

m h　dom mon dow　　command

#两分钟爬取一次每日各时段的详细天气信息

2 * * * * sudo sh /home/by/python_test/today_weather/cron.sh

#每隔一小时爬去一次未来七天的天气走向。

0 * * * * sudo sh /home/by/python_test/weather/cron.sh

4.3 数据存储

数据库存储着整个系统的全部天气数据、用户登录信息、用户详细信息和一份全国城市天气代码对照表。

表 4.1　数据库结构表

表名	功能
city_code	全部的城市代码：省、市、地区、城市代码
register_city	所有已经被注册的城市的城市代码集合
user_info	每一个用户所注册的城市代码集合
user_login	用户的用户名与密码
weather7day	近七天的大致天气(仅有每天的最高温度与最低温度
weather7day_full	近七天的详细天气（按小时统计），包括今天的详细天气

表 4.1 为数据库中的全部表格及其功能含义。

下面详细介绍每一个表的设计：

City_code 表：

| city_code | CREATE TABLE `city_code` (
`province` varchar(20) DEFAULT NULL,
`city` varchar(20) DEFAULT NULL,
`area` varchar(20) DEFAULT NULL,
`city_code` varchar(20) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='全部的城市代码：省、市、地区、城市代码'

Register_city 表:

| register_city | CREATE TABLE `register_city` (
`city_code` varchar(20) NOT NULL,
PRIMARY KEY (`city_code`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COMMENT='所有已经被注册的城市的城市代码集合'

User_info 表：

| user_info | CREATE TABLE `user_info` (
`username` varchar(20) NOT NULL,
`city` varchar(20) NOT NULL,
PRIMARY KEY (`username`,`city`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COMMENT='每一个用户所注册的城市'

User_login 表：

| user_login | CREATE TABLE `user_login` (
`username` varchar(20) NOT NULL,
`password` varchar(20) NOT NULL,
PRIMARY KEY (`username`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COMMENT='用户的用户名与密码'

Weather7day 表:

| weather7day | CREATE TABLE `weather7day` (
`id` int(11) NOT NULL AUTO_INCREMENT,
`city_code` varchar(10) NOT NULL,

`weatherDate1` varchar(10) DEFAULT NULL,
`weatherDate2` varchar(10) DEFAULT NULL,
`weatherWea` varchar(10) NOT NULL,
`weatherTem1` varchar(10) NOT NULL,
`weatherTem2` varchar(10) NOT NULL,
`weatherWin` varchar(10) NOT NULL,
`updateTime` datetime NOT NULL,
PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=29 DEFAULT CHARSET=utf8 COMMENT='近七天的大致天气(仅有每天的最高温度与最低温度)'

Weather7day_full 表：

| weather7day_full | CREATE TABLE `weather7day_full` (
`id` int(11) NOT NULL AUTO_INCREMENT,
`city_code` varchar(10) NOT NULL,
`weatherDate` varchar(10) DEFAULT NULL,
`weatherWea` varchar(10) NOT NULL,
`weatherTem` varchar(10) NOT NULL,
`weatherWinf` varchar(10) NOT NULL,
`weatherWinl` varchar(10) NOT NULL,
`updateTime` datetime NOT NULL,
PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=161 DEFAULT CHARSET=utf8 COMMENT='近七天的详细天气（按小时统计），包括今天的详细天气'

4.4 服务器程序的设计实现

服务器程序使用 python 语言编写，是一个可以长期运行的服务器程序，当有客户端连接请求被发送过来的时候，服务器接手请求并对请求进行处理，根据请求的不同，服务器可以提供的服务分别有：用户登录验证，今日天气信息查询，未来七天天气走向查询，注册城市等功能。

图 4.14 是服务器与客户端的启动流程，服务器是长期运行的，客户端用完就可以结束，并在再次需要的时候启动。

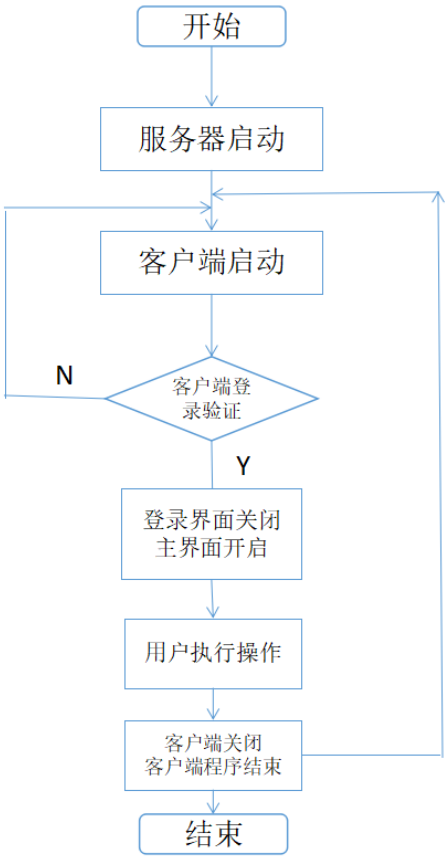

图 4. 14　服务器与客户端启动流程

4.4.1 使用网络套接字与 Client 通信

服务器使用 socket 与客户端通信，下面是服务器创建套接字，并监听指定端口是否有客户端连接，以及数据的发送与接收部分。

```
import socket
# Address
HOST = ""
PORT = 8955
# Configure socket
skt = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
#AF_INET 的含义是让套接字使用 IPV4 协议作为网络层协议
#SOCK_STREAM 的含义让套接字使用 TCP 协议作为传输层协议
skt.setsockopt(socket.SOL_SOCKET,socket.SO_REUSEADDR,1)
#SO_REUSEADDR 是让处于 TIME_WAIT 状态的端口再次被使用
#套接字绑定端口
skt.bind((HOST, PORT))
```

```
#最大连接数为 1024
Skt.listen(1024)
#服务器开始监听指定端口
conn, addr = s.accept()
#若连接已经建立，通过套接字接收客户端发送过来的消息
request       = conn.recv(1024)
#通过套接字发送回复到客户端
conn.sendall(reply)
#关闭套接字
conn.close()
```

4.4.2 Server 验证登录用户合法性

服务器对客户端传递的参数进行判别，并掉调用不同函数进行处理后，将结果返回给客户端，下面是服务器的判别参数的模块：

```
# 接收和确认连接
while 1:
    conn, addr = s.accept()#阻塞并监听用户连接请求
    request       = conn.recv(1024)#最大连接数为 1024
    print 'Connected by', addr
    if request == 'user_info':
        reply = select_user_info(request)
    elif request == 'user_login':
        reply = select_user_login(request)
    elif request == 'weather7day':
        reply = select_weather7day(request)
    elif request == 'weather7day_full':
        reply = select_weather7day_full(request)
    conn.sendall(reply)#发送信息
    conn.close()#关闭连接
```

用户登录验证模块，根据用户端传递的参数，验证用户名与密码是否匹配，若用户名存在且密码匹配，则返回”yes”，否则返回”no”。图 4.15 是其实现。

```
def select_user_login(request):
    connect = None
    #从参数中取出用户名密码
    username = list(request.split(‘,’))[0]
    password = list(request.split(‘,’))[1]
    try:
        connect=MySQLdb.connect(user=dbuser, passwd=dbpass, db=dbname, host=dbhost, charset="utf8", use_unicode=True)
        cur=connect.cursor()
        cur.execute(“select password from user_login where username = ‘%s’;”,% username)
        reply = 'no '
        result = cur.fetchone()
        if not result:
            if password == result[0]:
                reply=’yes’
        if connect:
            cur.close()
            connect.close()
        return reply
    except MySQLdb.Error, e:
        print "Error %d: %s" % (e.args[0], e.args[1])
        sys.exit()
```

图 4.15　用户登录校验函数的实现

4.4.3 后端数据分析与查询

Scrapy 爬虫程序会定时查询天气数据，并更新到 mysql 数据库中，总是将最新的天气数据保存到文件中。每当用户端发来天气查询的请求，服务器都会将文件内容作为结果返回给客户端。

图 4.16 是 server 程序中处理天气查询的部分的实现：

```
def select_weather7day_full():
    f = None#初始化f为None
    try:
        #打开保存着数据信息的文件
        f = open('/var/lib/mysql-files/weather7day_full.txt')
        print('weather7day_full.txt is opened')
        reply = f.read()
        #读取文件信息到字符串里
        print("file content:\n%s"%reply)
        return reply
        #将结果值返回
    except Exception as err:#异常处理
        print("File operation failed:" + str(err))
        return ' '
    finally:
        if f:
            f.close()#读取文件失败，关闭文件描述符f
            print('weather7day_full.txt is closed')
    pass
```

图 4.16　select_weather7day_full()函数的实现

以及通过定时脚本任务来更新最新的数据到文件中，下面是脚本 cron.sh 的内容：

```
#! /bin/sh export PATH=$PATH:/usr/local/bin
#若旧数据文件存在，则删除掉
file="/var/lib/mysql-files/weather7day_full.txt"
if [ -f "$file" ];then
  rm -f "$file"
fi
#进入到指定目录下，执行爬虫程序，抓取数据，并将标准输出和标准错误都保留到日志文件里
cd /home/by/python_test/today_weather/
#rm weather.log
nohup scrapy crawl weather >> weather.log 2>&1
#执行mysql命令，登录mysql并将最新的天气数据保存到文件里
sql_txt="select * from weather7day_full  into outfile
        '/var/lib/mysql-files/weather7day_full.txt' lines terminated by '\n';"
mysql -uadminuser -p123456 bs_db -e "${sql_txt}"
```

图 4.17　cron.sh 脚本

4.4.4 使用 IO 多路复用提升 Server 性能

服务器程序通过使用 I/O 多路复用函数，可以提升处理性能。图 4.18 是其实现：

```
r_list = [s,]
num = 0
while True:
    rl,wl,error = select.select(r_list, [], [])
    num += 1
    print('--------counts is %s'%num)
    print("--------rl's length is %s"%len(rl))
    for fd in rl:
        if fd == s:
            conn, addr = fd.accept()
            r_list.append(conn)
            request = conn.recv(1024)
            print 'Connected by', addr
            print request
            reply = "connect to server succeed!"
            conn.sendall(reply)
        else:
            try:
                request = conn.recv(1024)
                print 'requests: ',request
                if request == 'user_info':
                    reply = select_user_info()
                elif request == 'user_login':
                    reply = select_user_login()
                elif request == 'weather7day':
                     reply = select_weather7day()
                elif request == 'weather7day_full':
                     reply = select_weather7day_full()
                conn.sendall(reply)

                #fd.sendall('second ...,,,,;;;')
            except ConnectionAbortedError:
                r_list.remove(fd)
                #sys.exit()
    conn.close()
```

图 4.18　使用 select 优化 server

4.5 Client 的设计与实现

客户端使用 Qt 进行开发，Qt 是基于 C++的一套 GUI 开发库，同时拥有非常

丰富的接口，包括网络、线程、绘图库等等。本课题使用 Qt 开发客户端，用户从客户端登录，进入主界面，并可以从主界面进行城市注册、天气查询等操作。

4.5.1 使用 Qt 进行 GUI 开发

使用 Qt 进行客户端的开发，客户端界面有两部分，包括用户登录界面和登陆成功之后的主界面，QT 提供了 Qt Designer 工具，能很方便的通过鼠标拖拽的方式去绘制界面。绘制完后自动生成一个图形界面的.h 文件（如 ui_mainwindow.h），其中含有一个自动生成的 Ui_MainWindow 类，这个类中核心的函数是 setupUi。登录界面的类是公有继承自 QDidalog 的 login 类，界面的切换是在 main 函数中进行的：

```
intmain(intargc,char*argv[])
{
QApplicationa(argc,argv);
Loginlog;   //创建登录界面对象
if(log.exec()==QDialog::Accepted){
log.close();
MainWindoww;
w.show();
Returna.exec();
}
else
return0;
}
```

启动 log 并等待用户输入，当用户点击登录按钮，而且用户名密码匹配成功的时候，login.cpp 会调用 accept 函数，返回一个值 Accepted，main 函数接收到这个值，会将 log 关闭，创建一个 MainWindow 对象并将其显示，然后创建一个事件循环。

4.5.2 用户登录模块设计与实现

图 4.19　客户端登录界面

如图 4.19，用户登录模块主要是通过前台输入用户名密码，然后建立 socket 连接服务器，将用户名密码通过字符串的形式发送给服务器，服务器在后端进行用户名密码的校验，若校验成功，则返回 yes，客户端接收到 yes，会通过验证；若收到 no，则弹窗“用户名或密码错误”。

```
void login::on_pushButton_clicked()
{
    QString user=ui->lineEdit->text();
    QString passwd=ui->lineEdit_2->text();
    if( !user_exist_true(user,passwd) ){
        QMessageBox::warning(this,tr("oh"),tr("User name or Password Wrong"),QMessageBox::Yes);
        ui->lineEdit_2->clear();
    }
    else{
        qDebug()<<"OK"<<endl;
        accept();
    }
}
```

4.5.3 数据分析与曲线绘制

如图 4.20，是用户主界面的展示效果，可以查看未来七天每一天的详细天气信息：

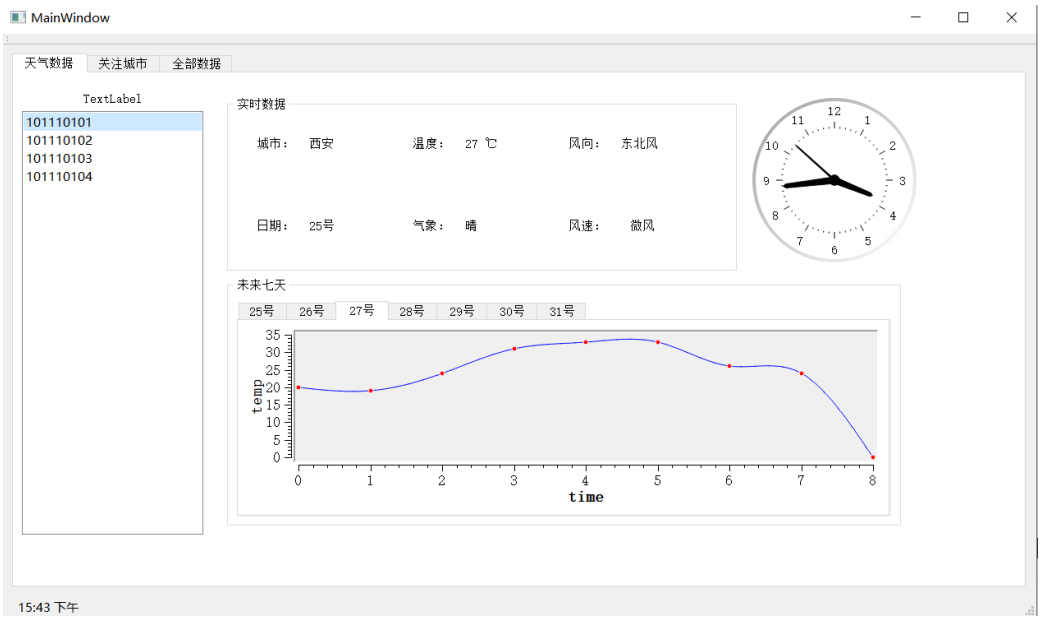

图 4.20　数据分析与展示界面

```
QVector<double>xs;
QVector<double>ys;
save_value(xs,ys);
QwtPlotCurve*curve=newQwtPlotCurve("Temp");
curve->setSamples(xs,ys);

curve->setPen(QPen(Qt::blue));//线条颜色设置为蓝色
curve->setRenderHint(QwtPlotItem::RenderAntialiased);//启用抗锯齿
curve->setStyle(QwtPlotCurve::Lines);//直线形式
curve->setCurveAttribute(QwtPlotCurve::Fitted,true);//是曲线更光滑
curve->setPen(QPen(Qt::blue));//设置画笔

//设置样本点的颜色、大小,Ellipse 椭圆或圆，brush 填充内部颜色，QPen 节点的轮廓颜色
QwtSymbol*symbol=newQwtSymbol(QwtSymbol::Ellipse,
QBrush(Qt::red),QPen(Qt::white,1),QSize(6,6));
//添加样本点形状
curve->setSymbol(symbol);
//ui->qwtPlot_5->show();
curve->attach(ui->qwtPlot_5);
ui->qwtPlot->replot();
```

结　论

本课题的论文凭借 Python 丰富的标准库和快速开发的特点，设计了一种基于中国天气网的网络爬虫程序，能为用户带来快速获取指定城市天气的使用体验。并通过 C/S 模式，将程序架构分为两部分：客户端与服务器，服务器上运行爬虫程序、server 程序、数据库管理工具等，客户端运行一个由 Qt 开发带界面的 client 程序。这些模块属于耦合度很低，但通过 Socket 网络通信，又有机的结合起来，成功实现了预期的功能。

作品最终得展现形式是一套完整的前台后台系统，用户通过在客户端登录，可以进入主界面查看自己订阅的城市的天气信息，客户端通过 socket 与服务器进行通信，将请求发送到服务器，服务器后端运行了爬虫程序与数据库程序，服务器通过这些后端系统收集信息与反馈信息，可以与客户端非常好的交互。天气信息会通过很直观的曲线的方式展示，鼠标放到曲线图上即可查看详细信息，未来七天的数据都会有详细图像展示。若用户想要查看所有数据，也有详细查看数据的部分。软件的功能设计是比较完善的。

虽然我的论文作品不是非常成熟，仍旧有许多不恰当的地方，比如对 python 网络编程的理解不是十分深刻，只实现了 Select 而没有成功实现 epoll 模式的 Server 程序；又比如对 Mysql 数据库的键的关系不太了解，数据库表之间的字段没有能够用主键/外键的方式建立起关系与关联，但每一段代码都有我的劳动，这让我很欣慰。

这次毕业设计的经历也定会让我终身受益，我感受到做毕设是需要自己认真地去对待，是自己真正独立学习和研究的过程，没有学习就没有研究的能力，也不能完成这次毕设，这也为我以后的工作与生活带来宝贵的经验。

致　谢

首先感谢导师石薇老师，在我论文开题和写作过程中给予了悉心的指导，在我写作最为困难的时候对我提供了思路和方向。在导师严谨、负责的治学态度影响下，我认真的完成了本篇论文的每一部分，在此对辛苦付出的石老师道一声真挚的感谢！

同时也要感谢我的各位同学，在学习中互相分享和帮助，是我完成毕设论文的动力。

完成论文的过程中，本人虽参考了大量的文献，但水平有限或许存在一些错误。恳请各位专家老师指正，不尽感谢！

参考文献

[1] 文焱.浅谈计算机网络管理技术及其发展趋势[J].消费电子，2012

[2] 焦赛美.网络爬虫技术的研究[N].琼州学院学报，2011

[3] 杨瑞.一种分布式网络爬虫的设计与实现[N].江西师范大学学报：自然科学版，2013

[4] 云贵全.基于 Java 多线程技术网络聊天程序的设计与实现[J].软件，2012

[5] 徐亦璐.基于多线程的网络爬虫设计与实现[J].计算机光盘软件与应用，2011

[6] 李志强.浅析 TCP / IP 三次握手机制的安全性[N].邯郸职业技术学院学报，2012

[7] 朱晓霄.大数据时代索引员的使命[M].中国索引，2013

[8] RFCs.2004.Hypertext Transfer Protoco HTTP/1.1[EB/OL] http://www.faqs.org/rfcs/rfc/1945.html

[9] RFCs.2004.Domain names[EB/OL].http://www.faqs.org/rfcs/rfc/1034.html

[10] 王岩.搜索引擎中的网络爬虫技术的发展[J].电信快报，2008，10(10)

[11] Allan Heydon,Marc Najork,Mercato r.A scalable ,extensible Web Crawler[J] World Wide Web,1999,2(4):219-299