电子科技大学

UNIVERSITY OF ELECTRONIC SCIENCE AND TECHNOLOGY OF CHINA

专业学位硕士学位论文

MASTER THESIS FOR PROFESSIONAL DEGREE

论文题目 网络爬虫系统的研究与实现

专业学位类别 工程硕士

学号 201191020126

作者姓名 赵茉莉

指导教师 张扬 教授

DESIGN AND RESEARCH OF NETWORK SPIDER

A Master Thesis Submitted to

University of Electronic Science and Technology of China

Major: **Master of Engineering**

Author: **Zhao Moli**

Advisor: **Zhang Yang**

School : **School of Electronic Engineering**

独创性声明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。据我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得电子科技大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

作者签名：_____ 日期：2013 年 4 月 17 日

论文使用授权

本学位论文作者完全了解电子科技大学有关保留、使用学位论文的规定，有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许论文被查阅和借阅。本人授权电子科技大学可以将学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。

（保密的学位论文在解密后应遵守此规定）

作者签名：_____ 导师签名：_____

日期：2013 年 4 月 18 日

分类号 _____ 密级 _____

UDC [注1] _____

学　位　论　文

网络爬虫系统的研究与实现

（题名和副题名）

赵茉莉

（作者姓名）

指导教师姓名 张扬　教 授

电子科技大学　成　都

尚尊义　高　工

大连交通大学　大　连

（职务、职称、学位、单位名称及地址）

申请专业学位级别 硕士　专业学位类别 工 程 硕 士

工程领域名称 软　件　工　程

提交论文日期 2013 年 3 月　论文答辩日期 2013 年 6 月

学位授予单位和日期 电 子 科 技 大 学　2013 年 6 月 29 日

答辩委员会主席 _____

评阅人 _____

2013 年　　月　　日

注 1：注明《国际十进分类法 UDC》的类号

摘 要

在网络中，我们可以获取各种各样的资源，然而有效的信息搜索难度也是日益加大。要有效的完成信息搜索，则通过搜索引擎的构建就能够对该问题有效的解决。

所谓的多线程网络爬虫程序实际上是由根据宽度优先算法对指定 Web 页面完成搜索及解析任务，同时抓取搜索中获取的所有 URL，同时进行保存，此外，在互联网中，将 URL 视作新入口，完成持续爬行动作，并对后台程序自动执行。

对于网络爬虫而言，其主要应用到的技术包括 socket 套接字、HTTP 协议、正则表达式以及 windows 网络编程等技术，该网络爬虫实际上是能够在后台进行运行的，其初始的 URL 是配置文件，实现爬行主要依靠宽度优先算法完成，对目标 URL 网络程序进行保存，语言的实现主要应用了 C++语言，且调试环境为 VC6.0，对于常见搜索任务就可以执行。

本论文中，第一章节对以英特网为基础的搜索引擎具有的系统结构进行了详细的阐述，主体部分则是对引擎搜索器的设计和实现进行具体的描述，其中网络爬虫即该搜索器。笔者本论文中的主要工作量包括：

1、针对网络爬虫 SPIDER，分析了其体系的结构；

2、设计了系统核心功能模块部分；

3、对 My SQL 数据库进行实现；

4、完成队列管理，主要利用 URL 解析完成；

5、设计各函数模块；

6、测试网络爬虫的系统。

同时，针对网络爬虫进行设计以及实现的内容，对技术核心详细的阐明，也综合考虑了实现多线程网络爬虫代码，对系统实现了准确的说明，也更有助于读者的理解。

关键字：网络爬虫，广度优先，多线程，搜索引擎，网络爬虫，URL 抓取

ABSTRACT

Network resources are very rich, but how to effectively search for information is a difficult thing. Create a search engine is the best way to solve this problem.

Multi-threaded web crawler program is the first algorithm in accordance with the width from the specified Web page to parse, search, and to crawl each URL to search, save and new entrance on the Internet constantly crawling the URL is automaticallyRun the daemon.

Web crawler application socket socket, regular expression, the HTTP protocol, windows network programming and other related technology, the Web crawler is a run in the background to the configuration file as the initial URL, down to crawl to the breadth-first algorithm , save target URL of the network program in C + + language as the implementation language, and in VC6.0 debugging by ordinary users be able to perform web search task.

This thesis first details the system architecture of the Internet-based search engines, and then provides details on how to design and implement search engine search engine - Web crawler. Of the subject completed the following work:

1. Complete analysis of Web crawler SPIDER architecture;
2. To complete the design of the main function module;
3. My SQL database;
4. URL parsing queue management;
5. To achieve the design of the individual function module;
6. Carried out the testing of the system of network reptiles.

In addition, in the Design and Implementation of Web crawler chapters in addition to the detailed elaboration of the technical core combined with the realization of the multi-threaded web crawler code to illustrate, and easy to understand.

Key Words：SPIDER， Breadth First Search， multi-threads， Internet search engine， Network spider， URL Captures

目 录

第 1 章 概述

1.1 引言

近年来互联网的发展速度是迅猛的，在庞大的网络信息中，全部用户均可以通过一定的手段得到其想要获取的知识，包括生活、科技以及军事等方面的内容。众所周知，对于不同个体而言，其需要摄取的知识是不同的。就算是同一个人，他在特定时间和地点，所关心的话题也是存在差异的。该现象很大程度的增加了目标信息获取的难度，因此，人们就需要一个工具，则搜索引擎抓住了机会。通常情况下，搜索引擎能够对网上无数的网页进行搜索，同时能够对输入的关键词完成索引操作。搜索引擎构建对于搜集网页意义重大。在计算机技术中，高效和准确的获取目标内容亟需解决。从而为搜索引擎创造了土壤。网络爬虫在搜索引擎中是核心的组成部分，其也因引擎的出现，而开始被人们关注。目前搜索引擎和人类的需求表现出息息相关的联系，而这是什么呢[1]？

从广义角度出发，搜索引擎包括所有在网络中能够实现检索的系统或者是工具，也就是能够通过因特网对用户搜索请求进行响应，对相应的系统和技术实现有效的返回。

从狭义角度出发，在网络中，搜索引擎特指一种软件，其具有自动搜索的功能，其也可以被认为是一种人工方式，能够进行信息的检索，主要能够实现的功能包括采集万维网上的信息资源，并对得到的结果实现标引以及分析，同时，组织索引得到的信息，最终能够形成一个数据库，其作为信息服务系统能够采取网站方式实现网络用户的有效检索。

简而言之，搜索引擎实质上为信息检索系统，该系统的存在环境为 WWW 网络中。检索过程中主要包括类目录型及关键词两种检索方式。前者实际为收集互联网中的大量信息资源，目录的初步划分则根据收集到的信息类型进行的，然后对大的分层进行细化，进入到各小分层中搜索者就能够获取所需信息；后者通过逻辑组合实现关键词的输入，由关键词搜索引擎则能检索到对应资源具体地址，最终能够以相关规则为依据，将和关键字匹配的网址及链接都向用户反馈。

对于搜索引擎而言，其实质上为网站，而该网站的主要功能就是能够实现信息检索服务的供给，其之所以能够帮助人们找到所需信息，主要是依靠编写程序

能够归类 INTERNET 中全部信息实现的。在互联网中，信息增长的趋势为几何级数，搜索引擎进行网站的搜索主要是依靠 spider 程序完成的，一页一页的进行，同时对搜索到的相关超级链接内容收录到同一个数据库，从而用户能够查询网站。

网络爬虫程序即 Bot 程序，其具有较强的专业性。主要功能包括对众多 Web 页面实现有效的查询。执行环节的起点则是简单 Web 页面，随后要实现对其他页面进行访问主要是依靠超链接完成的，重复以上的操作，便能够对全部的页面都进行检索和扫描，从而获取所需信息。

爬虫程序能够对网页实现自动的获取。其在搜索引擎中是至关重要的，其能够帮助引擎实现网络中相关网页的下载功能。该程序采取的实现策略以及运行效率如何，对搜索结果产生的影响都是显著的。各搜索引擎以搜索需求的不同，在搜索过程中选用最佳爬行策略。如果选择的爬虫程序是优秀且高效的，则搜索信息就能够做到及时和准确。

对于网络爬虫实现，主要内容及困难具体为：实现多线程；分配临界资源；选择及实现遍历 web 图策略；选择及实现存储的数据结构 [10]。

本文中实现多线程爬虫程序主要是利用 C#语言完成的，而得到的程序是以广度优先遍历算法为基础的。爬虫程序实现后，就相当于对某站点能够定点的搜索其包含的 URLs，若搜索要求其他的信息，则一方面要对 URLs 进行解析，且相应的信息的获取也应是同时进行的。

1.2 网络爬虫的研究背景与意义

搜索引擎实际上为信息检索系统，该系统所处的环境为 WWW 网络中。其工作方式主要包括分类目录及关键词为基础的两种检索形式。前者实际是互联网中的大量信息资源，目录的初步划分则根据收集到的信息类型进行的，然后对大的分层进行细化，进入到各小分层中搜索者就能够获取所需信息；后者通过逻辑组合实现关键词的输入，由关键词搜索引擎则能检索到对应资源具体地址，最终能够以相关规则为依据，将和关键字匹配的网址及链接都向用户反馈。[9]。

本论文中主要以搜索引擎搜索器为研究对象，主要针对其设计与实现过程进行详细的阐述，同时针对 Spider 程序技术重点和概念都能详细的说明，描述了算法，后台运行也是通过 C++语言完成的。

爬虫程序能够对网页自动的获取。其在搜索引擎中主要的功能是负责因特网中的网页的下载，且在引擎中至关重要。对于爬虫程序而言，其采取的实现策略

以及运行效率，对搜索结构都会产生直接影响。各搜索引擎由需求的不同，进行最佳爬行策略的选择。从而能够很好的完成网络中信息的。搜索爬虫程序保证优秀及高效，很大程度的能够保证获取的信息更准确也更及时。

1.3 网络爬虫国内外研究现状

网络初始阶段，网络爬虫就存在，目前对网络爬虫的研究成果也是繁多的。

最早的爬虫是 google 爬虫，该爬虫主要的功能包括针对各爬虫组件能够完成各异进程。维护单独 URL 服务器的过程中， URL 集合的下载则是必要的；网页的获取也能够由爬虫程序实现；在索引的进程中，能够对超链接以及关键字实现提取；进程的解决过程中，应该 URL 能实现相对路径向绝对路径的转换。上述各进程的通信主要是依靠文件系统。网络爬虫中获取多个进程主要是依靠网络存档雇员完成的，在一次性进行彻底的爬行过程中，对应了 64 个 hosts 。储存爬虫进程，主要在磁盘中，而储存来源则是非本地 URLs；爬行完成阶段中，通过大量的操作实现在各 host 种子 sets 中加入 URLs。

目前，市场上普遍使用的引擎包括 google 和百度等，这些引擎的爬虫程序技术都是保密的。而市面上的爬虫实现策略主要有：广度优先、Repetitive、定义以及深层次爬行等多种爬虫程序。同时， 估算 Web 页数量主要是以概率论为基础实现的，该抽样爬虫技术能够实现对互联网 Web 规模的评价； 通过包括爬行深度以及页面导入链接等分析方法，能够有效的对由程序下载无关 Web 页等在内的选择性的爬行程序实现限制[12]。

网络爬虫技术发展现状显示了，国际中 google 对 youtube 的收购是投入极大成本的，而收购的目的在于对视屏内容市场的获取。市场上众多的新兴公司对此业务范围也是有所涉及的，以 google 的发展为楷模，就应该投入到搜索引擎中。

搜索引擎的未来趋势为由技术就能够掌握互联网，提供给各大网站索引功能，有效结合计算机提供的算法以及人力手工完成的辅助编辑，因此，用户得到的结构相关性更大，同时，也使人类发现数学公式的单纯使用是不能够达到理想效果的，在检索过程中不应忽视人类智慧的重要作用，因此，网络爬虫程序是市场所迫切需要的。

1.4 研究内容

1.4.1 本课题解决的关键问题

本文中设计系统功能，主要是基于系统开发的基本思想及总体任务，网络爬虫能够实现对网络中相关信息的自动搜集。利用网络爬虫一方面可以将网络中的相关信息进行采集工引擎所用，另一方面，其起到定向信息采集器的作用，对部分特定信息和资源也能够实现定向采集，其中包括招聘以及租房等信息。

多次按爬虫程序的实现主要是利用程序语言来完成的，且该程序是以广度优先算法为基础的。针对网络爬虫的实现进行分析，发现存在以下的问题：采取广度优先爬行策略的原因，广度优先爬行的实现方法；选择多线程的原因，及其实现的方法；在实现系统时，存储数据；解析网页信息等内容。

利用对该爬虫程序的实现，对特定站点 URLs 能够有效搜集，同时在数据库中存储此 URLs。

1.4.2 本文的主要研究内容

主要研究内容为

（1）分析系统需求；

（2）阐明 SPIDER 体系结构；

（3）设计各功能模块；

（4）描述 SPIDER 工作流程；

（5）分析 SPIDER 的构造；

（6）选择和实现爬行策略；

（7）解析、抽取及保存 URL

1.5 论文章节安排

本论文主要由 7 部分组成。第一部分，详细的介绍了研究网络爬虫背景、现状以及意义。第二部分，针对网络爬虫开发所涉及到的技术进行详细的阐述。第三部分，分析了系统需求，主要研究对象为 Spider 体系结构以及各功能模块。第四部分，对本系统开发的环境进行了叙述，并设计了策略，主要包括分析爬行策略，以及完成 URL。第五部分，实现网络爬虫及模块等。第六部分，对程序进行测试。第七部分，总结全文，展望未来发展趋势，主要对本文的工作内容以及系

统具有的功能进行归纳总结，同时对本系统的缺陷以及未来系统应该如何的完善进行展望。

第 2 章 相关技术介绍

2.1 网络爬虫简介

2.1.1 Spider 的概念

网络爬虫（Network Spider）中的爬虫实际上为互联网比，其中 Spider 为网络，因此这个名词又能够解释为网络中爬行的爬虫。该程序完成网页的寻找主要是依靠网页链接地址实现的，由某个网站起，对网页相关内容进行读取，从中获取相关的链接地址，随后利用获取的链接地址便能够找到其他的网页，在这个循环过程中，能够将所需的网页都实现抓取。若整个互联网视为网站，则网络爬虫抓取网页能够基于原理实现。

搜索引擎要对网络中全部网页都实现抓取难度极大，由当前相关数据可知，最大容量引擎能够实现的网页抓取量也不过在 40%。而导致这一问题的主要原因就是抓取技术出现局限，对全部网页是不能够完全遍历的，大量的网页都不能够被连接到；另一方面，在储存以及处理方面都是存在一定漏洞的，若各页面 20K 平均大小计，计算得到的 100 亿网页容量为 100×2000G 字节，即便是成功的储存了，但是下载将面临更多的困难。即使能够存储，下载也存在问题。例如对于一台计算机而言，其下载速度若为 20K/S，要将全部网页下载则需要 340 台电脑同时进行也需要一年的时间。此外，因存在过大的数据量，极大程度的影响到了搜索的效率。所以，大量搜索引擎抓取过程中网络爬虫仅针对较重要的网页进行，该过程对网页重要性的评价主要是根据链接深度为重要指标[4][5]。

2.1.2 网络爬虫原理

网络爬虫实际上音译 Spider 得到，此外 Crawler，bots， robots 以及 wanderer 等都是其同义词。定义网络爬虫时，可从广义与狭义两个角度进行，从狭义角度看，该软件程序采取标准 http 协议对万维网信息空间的遍历依靠超链接与 Web 文档检索办法完成；广义角度出发，网络爬虫是对 Web 文档进行检索依靠 http 协议就能够实现。网络爬虫这一程序在网页的提取过程中表现出极强的功能，其在引擎中具有网页下载的功能，且在引擎中不可缺少。其实现某站点的访问主要是

依靠对站点 HTML 文档提出请求实现的。其实现 Web 空间的遍历，在各站点间移动，完成索引的自动建立，同时使其存至网页数据库。网络爬虫一旦到达某超级文本，其信息搜索及指向超级文本 U RL 地址的获取主要是依靠 HTML 语言标记结构实现的，对于用户干预是完全不依赖的。搜索过程中，网络爬虫的搜索策略选择也是按照一定标准进行的[3]。

2.1.3 网络爬虫搜索策略

1、宽度优先搜索算法

该算法也是广度优先搜索，其图搜索算法中是最简便的，该算法也为中多图的算法提供了关键的原型。其中典型的包括 Dijktra 单源最短路径以及 Prim 最小生成树等算法，他们与本算法的思想都是接近的。本算法对树节图的遍历主要是沿树宽度实现的，且该图算点法也是较传统的，一旦找到目标，立即终止算法。设计与实现该算法的过程都是较简单的，该搜索在盲目搜索范畴中。现在，该方法使用的主要目的是将网页的覆盖量最大程度的扩大。在爬虫的研究过程中，大量的研究都在聚焦爬虫中引入了宽度优先搜索策略。这一方法的基本理念提出，对应的一定链接的距离范围中，同初始 URL 与主题具有相关性有极大的概率。此外，结合宽度优先搜索和网页过滤，网页抓取主要依靠广度优先策略实现，然后过滤其中无关网页。该方法主要的缺点是抓取的网页不断增多的过程中，会下载大量无关网页，同时也会过滤掉这部分网页，也就降低了算法效率 [13][14]。

2、深度优先搜索

该方法采取的搜索策略具体为：深度优先搜索过程中，保持对图的搜索尽可能的“深”， 针对最新顶点， 若其以顶点作为起点，同时探测过程也是不存在边的情况下，进一步的探索则可沿此边完成。探索完结点 v 全部边后， 搜索则会回到始结点。该过程的终点为完成源结点能够到的全部结点。若仍然有未被发现结点的情况， 那么要以源结点为起点对上述过程重复进行，终点为节点全部被发现。对于多数的情况，深度优先使爬虫会出现陷入(trapped)的问题，因此，其不具有完备性以及最优化 [15]。

3、聚焦搜索策略

以第一代网络爬虫引擎为主要工具，进行网络抓取过程中，数量超过 1 000 000，对网页基本上不进行重新搜集，且索引刷新频率也是极低的。同时，检索花费的时间成本是极高的，通常情况下，操作时间都在 10 s 左右，也有比这更长的。

在网页信息增长过程中，其呈现指数级，且变化是动态的，上述通用搜索引擎具有巨大的局限性，科学技术进步和发展过程中，对相关网页资源需要定向抓取，因此聚焦爬虫找到了生存的契机。该爬虫采取的爬行策略为针对某特定主题页面的访问采取最好优先原则，对该主题相关的页面的获取更准确、高效且更快速， 页面抓取的指导思想为依靠内容与 Web 链接结构。也就对聚焦策略爬虫规则充分的应用。对于聚焦爬虫而言，其将对所下载页面自动评分， 随后由得分进行排序，最终实现队列的插入。 下一搜索要以最好形式进行，则采取的是分析完弹出队列第一页面，然后完成执行， 该策略实现爬虫的优先跟踪，通常跟踪的是存在链接至目标页面可能的相关页面。 对于网络爬虫搜索策略而言，其关键部分为对链接价值的评价， 也就是所说的链接价值计算办法， 各价值评价方法对链接价值实现计算，链接重要程度表现也是存在差异的，因此搜索策略也是不同的。 因页面中包含链接，但是一般页面的价值越高其所含有的链接的价值也是越高的，所以，评价链接价值过程中，也就是要评价页面的价值。该策略一般是专业搜索引擎应用的较多， 由于该搜索引擎仅对某个主题页面进行关注。也具有繁多的聚焦收索策略，包括：以内容评价搜索、链接结构评价、固学习以及语境图等四种主要的聚焦搜索。

各搜索策略优缺点是显著的，研究网络爬虫搜索策略相关内容，对于发展和应用搜索引擎意义是重大的。策略越好，就越能够对应规定时间中，完成相关页面的获取依靠最少网络、计算以及存储的资源消耗。所以，在网络爬虫的发展中，未来使用策略需要对链接价值预测不断的完善，包括准确性的提高，计算时空复杂度的降低，同时也要加大力度增强爬虫自适应性等，突破束缚，快速的发展[7]。

2.2 C#线程

2.2.1 线程概述

基本上所有的操作系统对线程定义都是支持的，进程即某程度中程序是相互隔离且运行是独立进行的。通常，上述的操作系统对多进程操作都能够提供很好的支持。多进程实际上为在系统中能够实现多个程序的同时运行。例，编写采取 Microsoft Word 时，一般也希望能够用 mp3 播放器对音乐进行播放，有时，也要实现 Word 便捷，此外计算机也要完成打印任务，此外，也可用 IE 完成 Flash 动画

的下载。而上述的操作都是同步的，也就是不需要有时间先后，上述工作都是同时的。

实际上，在 CPU 中，某时间点执行的程序数是唯一的。而 CPU 只是实现了对上述程序能够不断 “跳跃”的执行。应用过程中未察觉中断问题主要是由于人类感觉速度远远低于其跳跃的速度。人类感知以秒计。CPU 则以毫秒计，肉眼看其为连续的。

所以，尽管肉眼看到的同步操作，在计算机中，其某特定点运行程序都是唯一的，只有多个 CPU 的时候才不是这样的。

多线程对多进程操作概念不断的扩展，划分任务时，也逐渐的到了程序级别位置，因此，各程序能够看起来是同步完成的。各任务视作对应的一个线程，多线程程序实际上是能够对多个线程程序同时执行的 [2]。

类比多线程与多进程可得到两者间的主要差异：各进程均对应一组完整变量，但是线程只能实现同样数据的共享。

2.2.2 C#线程模型

众所周知计算机程序要实现执行需 CPU，程序代码及可存取数据三个要素。C#语言实现多线程机制主要依赖虚拟 CPU。即 C#程序中，其内部能够实现多个计算机的虚拟，各计算机都有一个线程与其对应，其具备相应 CPU，能够实现数据和代码的获取，所以，任务的执行能够独立完成，两者对代码及数据也是能够实现共用的。C#线程的实现主要依靠 C#。Thread 类，其能够将 CPU 功能进行虚拟，对获取的数据及代码都可以及时的接收、传递以及处理，同时能够保证运行控制的独立进行 [6]。

众所周知，各 C#应用程序中，其线程数应该是大于等于一的，该线程即主线程。

2.2.3 创建线程

线程创建方法一

利用 Thread 完成 C#中的线程创建。Thread 能够对线程完成创建的构造器种类众多，具体：

Thread()；线程创建。

Thread(Runnable target)；线程创建，对目标实现指定。

Thread(Runnable target，String name)；创建线程：名是 name；目标是 target。

Thread(String name)；创建线程：名是 name。

Thread(ThreadGroup group，Runnable target)；创建线程：属于 group 组，且以 target 为目标。

一般情况下，对 Thread 继承采取一个类，接下来是对其中 run()方法进行覆盖，因此该类从本质上转变为线程。

各线程种，要完成各个操作，主要是依据特定 Thread 对象中的 run()，其中方法 run()是实际上就是线程体。

start()方法的应用，能够实现线程转向 Runnable 的状态，其对该线程依靠线程调度器完成注册。对 start()方法进行调用，则对线程的执行也不是立即完成的，也就是说，其对应 Runnble 并非是 Running。

线程创建方法二

完成 Runnable 接口的实现，同时对接口定义唯一方法 run()也是要实现的，能够完成现场创建。Runnable 接口应用过程中，对于所需类对象是无法实现创建的，也就更不能执行了，但是运行则必定要利用 Thread 类实现。

由上述方法，若继承 Thread 类，那么该类能对 start 方法进行调用，即目标对象采取的是继承的 Thread 类当；但要对 Runnable 接口实现，那么该类为其他线程目标。

2.2.4 C#中的线程的生命周期

对于 C#线程而言，其由产生到消失的整个过程中，主要的状态包括：新建、运行、可运行，死亡及阻塞 5 种。而 Running 状态即运行状态中，是不在 C#定义线程状态中的，即 C#规范内的运行状态并不是真正意义的状态，其实质上是可运行状态的特殊情况。

采取 new 完成线程新建过程中，其是 New 状态，此时，线程对应的无操作。

对于线程调度程序而言，其对各线程实现调度主要是以调度策略为基础的，run 方法的调用，使已注册线程的执行存在可能，也能够使被调度的转到运行状态中。完成线程 run 方法后，则会抛弃线程，线程至死亡的状态。此时通过 restart 方法的调用对处于该状态的现场就无法重新开始，而对于死亡状态中的线程对象具有的各方法都是能够调用的。

调度情况直接影响线程调度程序，设置正在运行线程是 Runnable 状态，典型的包括若 Runnable 状态中进入的线程相比现行运行状态具有更高线程运行等级，则 Running 状态中，对于现行线程存在被踢出的可能，最终将返回到 Runnable 状态。

2.2.5 多线程同步

对于同时进行相互独立线程而言，其要求共享数据，同时对其他线程状态也是要综合考虑的，保证上述线程的同步，则机制的使用是必要的，从而能够对资源争用的冲突有效规避，也就预防了死锁。保证线程同步，C#中有多种机制。绝大多数的 C#同步的中心是对象锁定。对于 C#而言，其继承 Object 对象得到的对象的锁也是独立的。因为 C#各对象均来自继承 Object。因此，在 C#内，各对象锁都是一一对应的。因此线程共享更加的协调。对于 C#而言，要对线程同步进行实现，则也可以依靠 synchronized 的关键字实现。C#对程序内同步部分的定义主要是通过 synchronized 关键字实现。其实现主要操作包括定义各需要线程同步为临界区，该区域同时刻执行唯一线程。

2.2.6 URL 消重

2.2.6.1 URL 消重的意义

对于 SPIDER 系统而言，其运行时，页面的下载，每秒能够完成 10 个，其中绝大多数的 URL 分析都是重复的，新 URL 数量极少。对于持续下载而言，也是有相当少新 URL 的，例新浪网中，1 天新 URL 共计约 10000。该状况与虚拟储存器管理是非常类似的。而虚拟储存器实际功能包括请求调入以及置换等，该储存器系统扩充内存的容量主要是由逻辑角度出发的。核心的部分是实现了要启动作业，仅需将部分页和段加入即可，运行作业时，内存内不存在所需页和段的情况中，则必定存在请求的调入，但是，外存页和段对内部暂不运行页面实现置换。

对于 URL 而言，其具有巨大的消重工作量。例如在新浪新闻页面中，各页面为 50-60k 的大小，各页 URL 数量是 90-100，若以 10 页/s 的速度仅需下载，则对应的 URL 排重操作次数是 900-1000，各排重操作的查询目标为 URL 库，其包含几百万甚至几千万内容。在数据库系统中，该操作是十分恐怖的，理论角度出发，存储系统需磁盘 I/O 动作的情况下，这类查询需求是很难实现的。

2.2.6.2 网络爬虫 URL 去重储存库设计

对于爬虫而言，其在启动环节中，对于相同网页，下载过程则最好是不重复多次的，这是由于若出现重复，则无疑对 CPU 造成浪费，此外，对搜索引擎系统而言，其负荷也极大程度的被增加。针对此问题，若想有效的解决，则对相关超链接要进行应用，保证了待下载的 URL 是一次性的，则网页也就不会出现重复。

易知对于搜索引擎系统而言，以全局为起点完成专门检测的构建，其中任意 URL 相对网页文件是否已经下载，对已下载 URL 储存到相应的数据库中，此方法便是有效的。要深入的对研究爬虫高效工作的相关问题，即提高工作的效率。要有效的避免重复， URL 存储库构建则是必要的前提，同时检测已被下载 URL 时，对内存进行加载操作，检测内存速度必定要高于检测磁盘。本文由简起，逐步深入和优化，最终获取最佳解决方案。

2.2.6.3 基于磁盘的顺序存储

该环节即以已下载 URL 为对象按照顺序最终完成储存。储存已下载 URL 至磁盘记事本文件。因此爬虫线程进行下载前，都需要检索磁盘中此文件以确定不重复，若是不重复，下一步就是编写该 URL 到记事本末行，要不就对下载操作进行撤销。

该方式目前已经被淘汰了，而该思想体现的十分直观。假设 100 亿已下载网页，必定对应相等数量的链接，即相应的记事本文件中储存 URL 也是同等数量，此外，加上绝大多数的 URL 字符串也具有较大的长度，对空间需求就打，因此极大程度的降低了效率，因此应舍弃该方案 [17]。

2.3 URL 类访问网络

若仅仅是希望请求文件只针对特定 URI 进行，那么具有极简单的使用过程。其中，NET 基类对应的为 System。Net。WebClient。该类属于高层类范畴，其主要功能对操作 1 个或者 2 个命令情况中，能有效的执行。对于.NET Framework 而言，其现在能够支持的 URI 主要包括开头是 http：、file：以及 https：的。

System 包含了 WebClient。空间基于 Net 命名的，利用该类能够对 Http 请求创建，并保证过程的便捷，此外能够得到相应的返回内容[8]。

2.4 爬行策略浅析

2.4.1 宽度或深度优先搜索策略

对于搜索引擎而言，其最早的网络爬虫主要特点有：以传统图算法为基础，主要包括对 Web 索引通过宽度优先或者是深度优先算法实现的，其种子集合通常是核心 U RL 集充当的，由该算法通过递归实现跟踪的超链，通常情况下能够与其他页面实现连接，但是，一般对页面内容无要求，由于最终目标要实现在整个 Web 中该跟踪均可以实现覆盖。该策略通的适用范围是通用搜索引擎，主要原因在于这类引擎要求网页越多效果越好，也不存在特定要求。

1、宽度优先搜索算法

该算法也是广度优先搜索，其图搜索算法中使最简便的，该算法也为中多图的算法提供了关键的原型。其中典型的包括 Dijktra 单源最短路径以及 Prim 最小生成树等算法，他们与本算法的思想都是接近的。本算法对树节图的遍历主要是沿树宽度实现的，且该图算点法也是较传统的，一旦找到目标，立即终止算法。设计与实现该算法的过程都是较简单的，该搜索在盲目搜索范畴中。现在，该方法使用的主要目的是将网页的覆盖量最大程度的扩大。在爬虫的研究过程中，大量的研究都在聚焦爬虫中引入了宽度优先搜索策略。这一方法的基本理念提出，对应的一定链接的距离范围中，同初始 URL 与主题具有相关性有极大的概率。此外，结合宽度优先搜索和网页过滤，网页抓取主要依靠广度优先策略实现，然后过滤其中无关网页。该方法主要的缺点是抓取的网页不断增多的过程中，会下载大量无关网页，同时也会过滤掉这部分网页，也就降低了算法效率 [21]。

2、深度优先搜索

该方法采取的搜索策略具体为：深度优先搜索过程中，保持对图的搜索尽可能的“深”， 针对最新顶点， 若其以顶点作为起点，同时探测过程也是不存在边的情况下，进一步的探索则可沿此边完成。探索玩结点 v 全部边后， 搜索则会回到始结点。该过程的终点为完成源结点能够到的全部结点。若仍然有未被发现结点的情况， 那么要以源结点为起点对上述过程重复进行，终点为节点全部被发现。对于多数的情况，深度优先使爬虫会出现陷入(trapped)的问题，因此，其不具有完备性以及最优化。

2.4.2 聚焦搜索策略

以第一代网络爬虫引擎为主要工具，进行网络抓取过程中，数量超过 1 000 000，对网页基本上不进行重新搜集，且索引刷新频率也是极低的。同时，检索花

费的时间成本是极高的，通常情况下，操作时间都在 10 s 左右，也有比这更长的。在网页信息增长过程中，其呈现指数级，且变化时动态的，上述通用搜索引擎具有巨大的局限性，科学技术进步和发展过程中，对相关网页资源需要定向抓取，因此聚焦爬虫找到了生存的契机。该爬虫采取的爬行策略为针对某特定主题页面的访问采取最好优先原则，对该主题相关的页面的获取更准确、高效且更快速， 页面抓取的指导思想为依靠内容与 Web 链接结构[2]。

对于聚焦爬虫而言，其将对所下载页面自动评分， 随后由得分进行排序， 最终实现队列的插入。 下一搜索要以最好形式进行，则采取的是分析完弹出队列第一页面，然后完成执行， 该策略实现爬虫的优先跟踪，通常跟踪的是存在链接至目标页面可能的相关页面。 对于网络爬虫搜索策略而言，其关键部分为对链接价值的评价， 也就是所说的链接价值计算办法， 各价值评价方法对链接价值实现计算，链接重要程度表现也是存在差异的，因此搜索策略也是不同的。 因页面中包含链接，但是一般页面的价值越高其所含有的链接的价值也是越高的，所以，评价链接价值过程中，也就是要评价页面的价值。该策略一般是专业搜索引擎应用的较多， 由于该搜索引擎仅对某个主题页面进行关注。

2.4.3 基于内容评价的搜索策略

该策略 [3,4]对链接价值的评价主要是以主题和链接文本存在的相似度实现的，同时基于此，能够对搜索策略进行确定，所谓的链接文本实际上为填在链接附近说明性文字，也包括与链接 U RL 有关文字信息，评价相似度公式为：

$$sim(d_i, d_j) = \Sigma_{mk=1} w_{ik} \times w_{jk} / (\Sigma_{mk=1} w_{2ik}) (\Sigma_{mk=1} w_{2jk}) \quad (2\text{-}1)$$

式中，di—新文相应的本特征向量， d j —第 j 类对应的中心向量，m —特征向量维数，wk —向量第 K 维。因 Web 页面和传统文本具有较大的差异，该文档为半结构化，其中所有的信息都是具有管联的。页面链接对页面和页面的关系进行了指示，所以，部分学者已链接结构评价为基础，完成了链接价值方法的研究。

2.4.4 基于链接结构评价的搜索策略

该策略具体内容为：对链接重要性的确定主要是利用分析 Web 页面的相互关系实现的，对访问的顺序也提出了有效的方法。一般情况下，价值较高的页面包括入链或者出链都较多。典型的算法有 PageRank 及 Hits 等。

1、PageRank 算法

采取该策略的引擎以 Google 为代表, 即其网址为 http ：//www。Google。com ，该引擎开发的它独创的 PageRank 算法是具有独特性的，其也称作链接评价体系，通常网页链接数量就决定了网页重要性， 尤其是公认较重要的链接数。对于 PageRank 算法而言，其最早是在 Google 搜索引擎中应用的，检索过程排序查询了相关信息结果 [5]，近期，该算法在应用网络爬虫过程中，评价链接重要性环节内，该算法将页面价值表征为相应的 PageRank 值， 对应页面 p 的 PageRank 值可表征为 PR (p) ， 其对应迭代公式为：

PR (p) = C 1T+ (1 - C) Σ C∈in (p)PR (p)ûou t (C) û (2-2)

式中，T —页面总量， C< 1 代表阻尼的常数因子，in (p) —全部指向 p 页面集合，out(C)对应 C 页面出链集合。以 PageRank 算法为基础的网络爬虫，搜索时，对页面价值实现确定主要是通过计算各已访问页面具有的 PageRank 值来，先访问的页面一般为具有较大 PageRank 值的。

2、HITS 算法

HITS 方法实现了对 Authority 及 Hub 定义，这两个定义都是至关重要的。其中 Authority 实际上为其他页面引用权威页面的次数，也就是入度值。对于网页而言，其具有越大的被引用数量，也就具有越大的 Authority 值；Web 页面对其他页面指向次数由 Hub 表征, 该值也就是出度值。此值越大, 则具有越高的 Hub 值。因为较高 Hub 值页面一般情况下能够完成指向权威页面链接的提供，所以，对某一主题页面能够隐含其权威性。HITS 算法是基于 HuböAuthority 方法构建的，Authority 实际上为页面被引用次数，即入度值。 Hub 实际上时 Web 页面指向页面次数， 即出度值。 具体算法为： 向以关键字匹配为基础的引擎提交查询 q。 由引擎实现大量网页的返回，这些网页将前 n 个视作根集， 表示为 S。将被 S 引用或者是将 S 引入的网页加入到 S 中，扩展 S 为更大集合 T。顶点集 V l 采取的是 T 内 Hub，顶点集 V 2 是权威网页，V1 至 V 2 间超链接即边集 E ， 产生的图是二分有向的，表示成 S G = (V 1，V 2， E)。在 V 1 范围内，任意顶点 v ， v 对应 Hub 值表示为 H (v)， V 2 顶点 u 的 Authority 值表示为 A (u)。以 H (v) = A (u) = 1 为起点，在 u 上采取的公式(1) 对 A (u)进行修改, 在 v 通过公式(2)对 H (v) 进行修改， 下一步是实现 A (u) 的规范，用 H (v)表示，对上述运算重复的进行，以 A (u)和 H (v) 收敛为终点[24][26]。

A (u) = Σ v： (v ， u) ∈EH (v) (1)

$$H(v) = \Sigma v : (v, u) \in EA(v) \quad (2) \qquad (2\text{-}3)$$

在式(1)，能够对网页中较好 Hub 指向进行反映，也就会增加权威值。在式(2)中，能够对网页指向权威页面进行充分的反映，即页面增加 Hub 值也增加。尽管以链接结构价为基础的搜索对链接结构与页面存在的关联综合的考虑，而对页面和主题关系却被忽视了，对于部分状况，搜索偏离主题现象也是时常发生的。同时，对于搜索环节而言，对包括 PageRank 值、Hub 权重以及 Authority 在内的值都是要重复计算的，对于计算复杂度而已，其也是在链接数和页面的增长过程中，采取指数级的方式实现增加[6]。

2.4.5 基于巩固学习的聚焦搜索

近期，研究 Web 信息资源分布后，得到以下结论，绝大多数相同类型网站具有类似的构建方式，而网页若具有相同主题，则会具有相同的组织方式，部分的学者则希望在训练过程中引入巩固学习的方法，由上述相似点能够得到一定的经验，但是这些信息回报较好的情况是搜索与页面集距离较远的情况，但是上述策略，对于该状况时方向确定难度较大。对于巩固学习模型而言，通常情况下，网络爬虫的访问针对相当多无关页面后，就会得到存在关联性的页面，这就是所谓的未来回报，预测未来回报的值就是未来的回报价值，由 Q 价值表征。该方法关键点在于对链接 Q 价值计算的有效方法进行学习，以未来回报价值为基础，对搜索方向进行确定。该搜索策略目前效率极低，对训练负担极大程度的增加。

2.4.6 基于语境图的聚焦搜索

以巩固学习为基础的网络爬虫，搜索方向的确定能够依靠链接 Q 价值计算实现的，而其对距离目标页面是不能够估计的。所以，Diligent 等研究出以“语境图”为基础的搜索策略，该策略主要是完成典型页面 web“语境图”的构建，从而对目标页面距离实现有效的估计，较近页面被访问时间也较靠前[7]，以“语境图”为基础的搜索策略，对通用搜索引擎“语境图”构建是要借鉴的，但是最终得到的结果也不全能代表真实情况，所以，该方式局限性较大。

2.5 正则表达式

对于网络爬虫程序而言，其实现对网页的抓取后，针对 HTML 页面，要提取出全部 URL 连接，因此在对其他标签实现区分时，正则表达式是最有效的途径。

简而言之，正则表达式主要功能是实现模式匹配以及模式替换。其能够实现匹配模式构建中应用特殊字符即可，接下来比较匹配模式和程序输入、数据文件和 WEB 页面表单输入等，程序的执行则以其中是不是含有匹配模式为依据。

例在用户输入邮件地址格式准确性的验证过程中，正则表达式是有效和普遍使用的方法。若利用该方法判断出格式准确，则相关表单的处理正常进行，若果不匹配，则会提示，要求其重新输入。因此，在 WEB 应用过程中正则表达式对逻辑判断意义重大。

本文实现网络爬虫过程中，对 URL 的判断依靠正则表达式实现。

2.5.1 正则表达式应用分析

正则表达式（RE）这一公式实际上时由按照特定模式匹配得到的一类字符串。

支持该公式的主要包括文本编辑软件、脚本工具以及类库等等，同时包括 Microsoft 中 Visual C++存在的交互式 IDE 对其也逐渐的形成支持。对于正则表达式而言，其主要的组成包括普通及元字符两部分。前者一般有字母以及数字，但是后者就会具备特殊含义，对于较简单状况而言，正则表达式实际为普通查找串。

以正则表达式的"testing"为例，其中不存在任何的元字符，对于"testing"以及"123testing"等期都是能够匹配的，却对"Testing"不匹配。

2.5.2 正则表达式的元字符分析

常见正则表达式元字符解释如下表。

表 2-1 正则表达式的元字符

元字符	描述
。	任何单个字符的匹配。
$	行结束符的匹配。
^	一行匹配的开始。
*	0 或多个在其之前字符的匹配。
\	引用符，匹配中，元字符视为普通字符。

[ ] [c1-c2] [^c1-c2]	对括号内任何字符进行匹配。具体包括正则表达式 r[aou]t 对 rat、rut 及 rot 都能匹配，除去 ret。括号中对字符区间制定通过连字符实现
\< \>	匹配词（word）开始（\<）以及结束（\>）。
\(\)	定义 \(和 \)间表达式是“组”（group），同时保存匹配该表达式字符至临时区域，引用则可采取\1 到\9。
|	逻辑“或”（Or）运算两匹配条件。
+	1 或多个其之前字符的匹配。
？	0 或 1 个之前字符的匹配。
\{i\} \{i，j\}	对指定数目字符进行匹配，该字符的定义先于其之前表达式。

2.6 本章小结

本章对网络爬虫相关概念进行介绍，针对其相关特点也做出阐述，基于上述研究，为公司发展设计找到启发点，接下来则对网络爬虫过程中所需的技术进行详细的描述，详细说明网络爬虫各部分组成。针对网络爬虫设计程序的构建，提出设计构思，得到系统应具有功能。

第 3 章 系统需求分析及模块设计

3.1 系统需求分析

对于 SPIDER 而言，其需要得到的对象，实际上就是网络中无数的网页，各个网页之间的联系也采取超链接形式实现，各网页和超链接都是一一对应的，把这一关系叫做统一资源定位符（URL）。假定网络为图 M(V，E)，其网页构成节点集用 V 表示，节点集和节点集存在的链接，实现了边集 E 的构建，SPIDER 对图 M 的遍历的起点是某节点，路线是其中的一个边，对各节点 Vi 进行访问同时完成相应处理。

要实现以上的目睹，设计 SPIDER 应是多线程，网络上工作主要是协同 A 个线程并发进行，这样就是先遍历的时间最短。而网页数目过大，若任 SPIDER 程序搜索为无穷的，则程序基本上没有终止。因此，对 SPIDER 工作访问唯一站点是要限制的。站点若是大型的，其网页是具有有限数量的，所以， SPIDER 程序的结束时间也是有限的。

对于 SPIDER 程序而言，其实现网页的访问过程中，要处理的基本项包括：对网页中文本进行抽取；对网页 URL 进行抽取，同时要对站内或者是站外的 URL 进行区分。

3.2 SPIDER 体系结构

该爬虫程序包括任务执行端，数据服务端以及任务调度端三部分。

执行各个 SPIDER 过程中，与一个站点、线程下载都是相关联的，其中下载的页面是以 URL 链接为基础的，此外也要解析 Web 页面，从而获取站内的 URL ，也能够对新站点发现相应的 URL，持久化 URL 队列，使其能够存入数据库，所以，执行 SPIDER 任务时，其端点外完成 Down 掉后，续传断点依旧可以进行。

协调 SPIDER 客户端时，主要是利用 C#完成的，具体采用是 synchronized，对于数据服务端而言，提高 URL 的缓存时，也使系统处理速度更快。执行 SPIDER 时，调度任务时，对 URL 队列都是要实现维持的，对于任务执行端 URL 队列而言，对站内 URL 实现储存；任务调度端即 URL。该 URL 队列存在的操作较大，

主要针对 URL，包括查找、插入以及状态更新等等。若 SPIDERWeb 页面下载的速度是 50 页/秒，每 6s 产生的 URL 数是 2000 多[12]，所以，对上述 URL 队列储存仅依靠内存数据结构是不能实现的，且对于系统而言，内存空间也是不足的；但是数据库选取直接持久化的时，对数据库的查询及连接等操作量也是巨大的，也就会较大程度的降低系统效率。若选取 URL 压缩法，虽然能够缓解一部分平衡空间与时间应用带来的矛盾，然而对于较大规模的数据采集是不适用的。

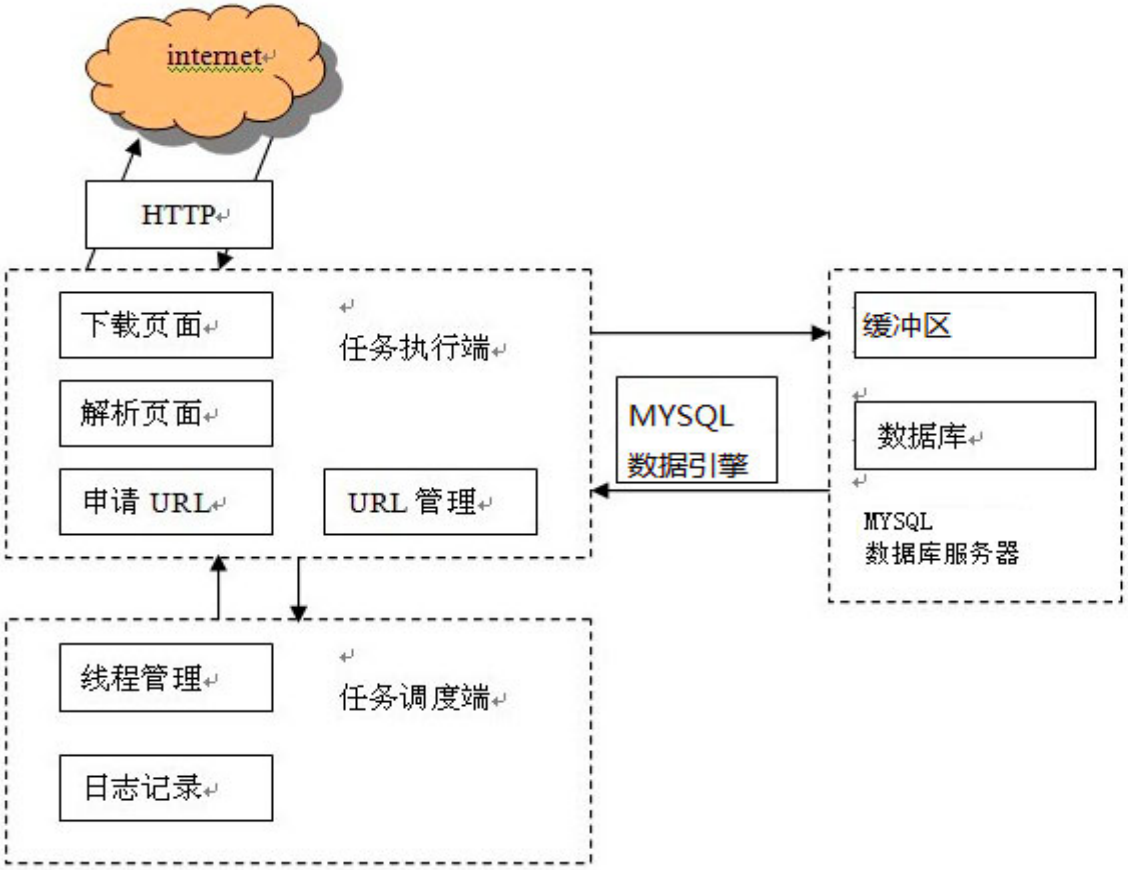

图 3-1 SPIDER 体系结

3.3 主要功能模块（类）的设计

MatchCollection 类：该类集合了选取迭代方式，能够实现正则表达式模式在字符串输入过程中的应用，从而能够实现匹配的成功进行。

Monitor 类：对对象可实现锁定功能，各线程操作各对象前，都需要先锁定。采取对象锁机制能够对可能存在的混乱情形进行有效避免，保证线程和时刻的一一对应。

3.4 SPIDER 工作过程

1、URL 等待队列中加入已知初始 URL。

2、爬虫线程的创建，并对其进行启动

3、各爬虫线程任务 URL 的获取都来源于 URL 等待队列。但是以 URL 方式完成网页下载后，对网页进行解析，最后时限超链接 URLs 的获取。若 URL 是相对的地址，则要对其进行转换，只有绝对地址才被识别，接下来对站外 URLs 进行淘汰，主要包括错误的或不能够被解析的。最后对上述 URL 下载是否存在进行判断，若未下载，则将其归入 URL 等待队列中。

4、重复步骤 3，满足结束条件后终止。

3.5 本章小结

本章主要分析了网络爬虫系统的需求，针对 Spider，设计体系结构和功能模块部分进行详细的介绍。

第 4 章 系统分析与设计

4.1 SPIDER 构造分析

经过对相关资料的总结和分析，我认识到有两种途径可以实现对 SPIDER 程序的构造。他们分别是：

1、用递归的方式设计 SPIDER 程序设计；

2、通过对一个不是递归程序的编写，实现对访问网页列表的维护。

使用上述哪种方式实现对 SPIDER 的构造，达到能够访问大量的网页的目的，就必须考虑编程技巧和使用的线程方式。由于本系统在编程中使用的多线程方式，这就不能达到递归方式要求的把每个消息或者信息按照一定的队列或者顺序存储到堆栈中，然后在按照一定的顺序进行访问。并且递归 方式只能实现对信息的单独连续调用，不能实现多任务的同时调用。因此，递归一般用于单线程中，多线程不能实现和递归的完全兼容。在多线程中，每一个独立线程在空闲时都可以要求获取 URLs 提供的任务，然后多个线程把所有的任务排列在一个公用的 URL 队列中等待执行。

通过对相关文章的阅读和归纳，SPIDER 程序的执行过程可以概述为以下几个步骤：

第一，建立可以发送请求给 DNS Cache 的 URL 所有任务列表。也就是我们常说的初始化 URL 种子，实现 DNS Cache 这个媒介把具有任务列表的 URL 传送到 URL 中。

第二，基于网络传输协议，通过 DNS 实现对网页依据某种计算方式和排序方式进行搜索和分析，在进行完相关的搜索和分析后，获取可以利用的相关 URL，并将其放回到初始列表中，进而实现网络爬虫不断的运行和循环。

网络爬虫系统可以对服务器实现域名解析，在利用具有高速度读取和存储数据的 Cache 这个特殊的存储子系统可以实现对数据快速和经常性的访问。下面，我们对我们上面提到的能够比较详细概述单线程网络爬虫的基本特征的示意图，表示在下图中。如图 4-1 所示。

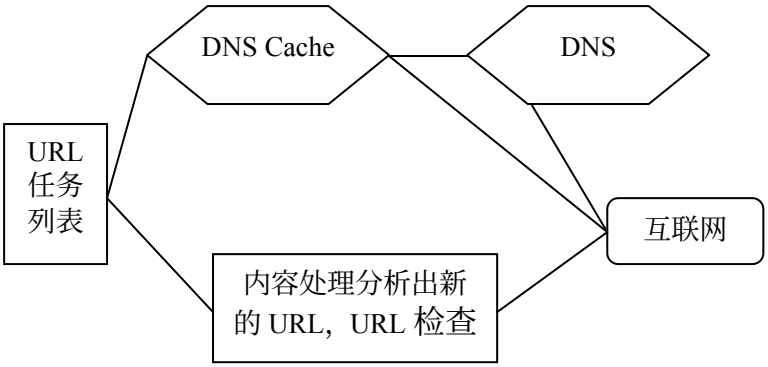

图 4-1 单线程网络爬虫模型图

4.2 爬行策略分析。

本论文是在广度优先计算法的基础上实现对本论文的爬虫网络的编程过程，其主要原因是，本论文想实现对一个站点下的多有页面进行访问，这个访问特点必须通过广度优先算法进行计算。利用广度算法实现我所设想的网络爬虫的模式如下：第一，对 HTML 中所有的 URLs 的初始值进行获取。第二，然后对上述过程进行反方向操作，实现对任务列表的和 HTML 中的 URLs，实现一一对应的。通过对上述两个过程进行不断的循环操作, 实现这一个层的全部 URLs 都能获得一一对应，然后在对下层的信息进行获取，以此类推，实现所有 URLs 的一一对应。这就是广度优先爬行网络的基本循环模式。

假如 a 为初始 URL 列表任务的代表，bcd 是基于在 a 的基础上取得的 3 个 URLs，efg 为基于在 b 的基础上取得 URLs，按照上述方法依次进行相似的计算，最后就会形成 abcdefghijklmnop 顺序的 URLs 模式。在 B 取得了 UOLs 后，不会马上对其进行解析，而是按照规则在解释完同层的 CD 后，然后对下一层进行解析。

具体的利用广度优先算法所设计的的等待队如图 4-2 所示。

图 4-2 列举了 URL 在不相同的时段中，其等待队列的存储情况。从左到有，所有的矩形图代表的意思依次为，对 URL 的初始化，并且在等待队列中添加 A，对 A 进行解析，其对应着获取器对应的 URLs 数值，bcd，然后删除 a；然后依次对 b 进行解析，获得其对应的 efg 这个 HTML 对应值，然后在对其删除，依次类推对所有的进行解析，我们从上面可以发现每一个 URL 初始列表中元素对应着三个 URLs 的值, 其他的也是一对三。上述的存储方式就是广度优先算法在爬行中的体现。。

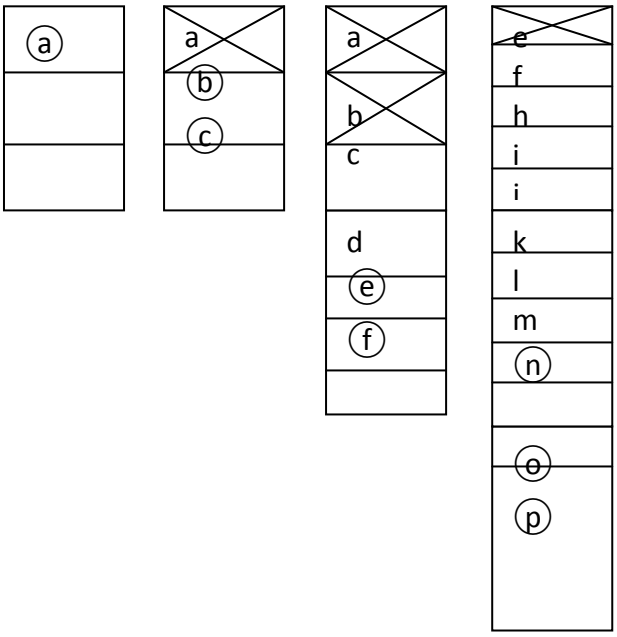

图 4-2 广度优先算法示意图

4.2.1 多线程爬虫模型分析

多线程爬行模型具体如下：再把 URL 任务列表的初始化的和种子放出到临时保存区后，然后利用多线程技术实现，每个请求有每个线程发送的同时，多个类似于爬虫触角的线程同时实现贵网络的访问，该访问是按照某种特定的算法和计算手段得到的，在获得网页的同时取得 URL 相对应的值返回到临时存储区。然后临时存储区把所有线程获得的 URL 值放入到任务列表中，作为下一个次操作的初始种子，这样一致的往返操作，就形成了爬虫程序的素有循环的示意过程，具体如下图 4-3 所示。

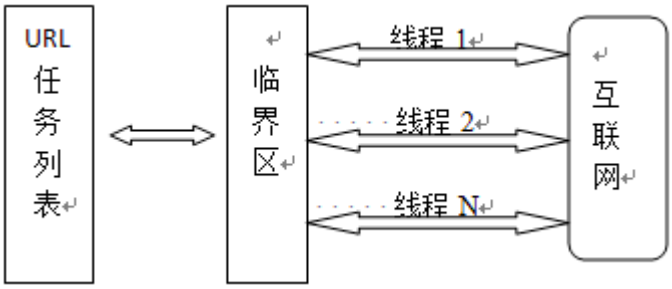

图 4-3 多线程网络爬虫示意图

4.2.2 爬虫集群模型分析

爬虫集群模型具体如下：第一，URL 的任务列表，通过一些方式把种子传输到 Spider 管理器中；第二，Spider 管理器按照一定的规则把所有的种子分配到一个 Spide 程序中，实现一一对应；第三，然后每个 Spider 程序作为类似于单线程操作的起始端，按照一定的方法和计算方式对对应的 URL 进行值得提取；第四，然后把提取的值返回到管理器，进过管理器按照一定规则的排列后，返回到任务列表中。第五，返回的任务列表里的 URL 值又作为下一个多线程操作的种子，然后继续进行与上述类似的操作，然后往返循环，直到整体爬虫程序运行完。其整个爬虫集群示意图如 4-4 所示。

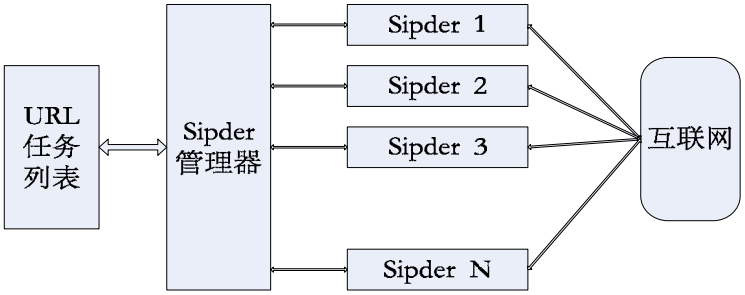

图 4-4 爬虫集群示意图

在上述的分析和探讨中，我们不难发现单线程模式的工作效率比较低、获取网页相关内容的过程耗费时间比较多，搜索的速度非常低，实现比较大型的编程功能非常复杂，而多线程就解决了单线程的不足，其使用效率和执行的速度都非常高，并且资源集成度非常高占据的资源比较少，因此，本文研究的重点就是多线程爬虫网络模型。

4.3 URL 抽取，解析和保存

4.3.1 URL 抽取

在基于 HTML 代码研究和分析的基础上，我们不难发现其实现网页之间互相转化这个工程是利用 href 标签这个关联通道实现的。因此，我们要实现对 HTML 代码中的 URLs 获取，就可以利用 href 标签的取得来实现。

我猜[20090531]5-31

火影忍者[331，页面上 303 既是]

通过对一些网站的编写程序的阅读我们发现，对于 href 标签出现的方式，有三种模式, 分别是 href=后加“URL”或者’URL’或者直接出现 URL。这样我们对 URL 的获取就可以通过对等号后面的相关表达式进行分析来取得，对于有双引号的应该去掉双引号，双引号之间的就是要获得的值，假如时单引号，单引号之间的就是要寻找的值，假如是上述第三种模式，我们就可以根据其等号后第一个空格或者>来断开表达式，第一个空格或者>就是我们要的值。

上述方法是比较常见的 href 标签的表达式，能实现对很多网页 URLs 的获取。但是, 有些网页有比较特殊的编程或 URLs 存储方式, 我们需要在编程时充分考虑。

4.3.2 URL 解析

通过上述方法获得的表达式或者字符串，可以理解为是 URL 的值，但是这些值能付作为地址排列到队列中呢，这当然不行，因为队列中需要的是绝对地址，而不是相对地址。所以，我们要对获取的相对地址进过一些手段和方式的处理，使其转变为绝对地址，然后加入到队列中。还有有些 URLs 的表达式比较诡异，坑内是一些文件的地址，对他们的解析就不能按照上述要求进行了，我们就得通过各种方式和手段，过滤到那些无用的东西，然后在假如到队列中。

4.3.3 URL 保存

由于第一个网页所获取的等待 URLs 非常多, 假如采取表格的形式对其进行存储，就会占据很大的内存，并且存取速度受到制约。因此，我们可以利用数据库实现对等待的 URLs 实现存储，然后建立两个缓存区实现对存取速度的提高的功能。上文所属的 URL 保存示意图如图 4-6 所示。

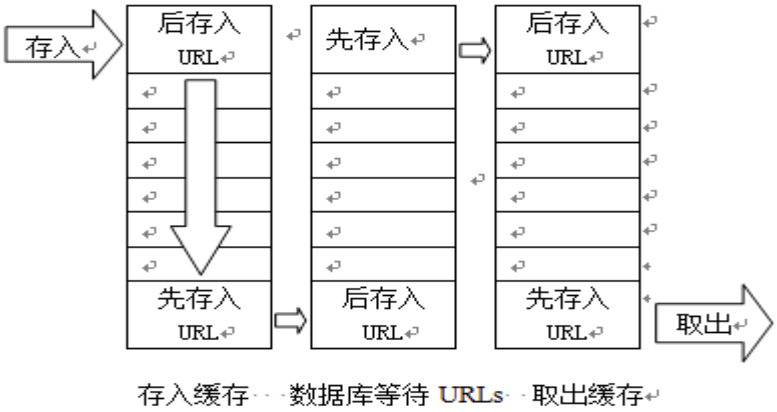

图 4-6 URL 保存示意图

由图示我们可以看出，等待队列被设计成为了了存入缓存、数据库和取出缓存三部分。这三部门的关系是这样的，当存入缓存中的数据出现溢出时，就把底层的以前存储的数据存储到数据库中，并且当取出缓存的数据别一一提取时，数据库等待的 URLs 就可以逐步一一加入到取出缓存中。这样既能保证数据存取的速度而且还能节约内存。

4.4 本章小结

本章节在介绍爬虫模型构造的基本方式的同时，对两种单线程和多线程的爬虫模式进行了比较详细分分析，并对 URL 的存取和保存相关的细节进行了总结和探讨，并举例进行了说明。

第 5 章 系统实现

5.1 实现工具

利用 XP 操作系统和 C#程序的编程语言结合 MYSQL 实现对系统的建立。

5.2 MYSQL 数据库的实现

对于 MYSQL 来说，他并具有广泛的用途和适用性，不仅仅可以适用于 java、php 等编程语言还可适用于 labview 等现代比较流行的图形化编程语言。他的使用非常简单，只要在电脑上安装 ODBC 这个 MYSQL 提供的驱动，就能实现和几乎所有编程语言和软件的兼容、混合使用。在 vs2008 编程软件中，我们可以使用类似于 JDBC 的名为：MySQLDriverCS.dll 文件包实现上述功能，具体操作如下：。把它添加到.net 的组件面板里[16]。

下面是创建一个数据库链接：

```
using MySQLDriverCS；
MySQLConnection conn = new MySQLConnection( new
MySQLConnectionString("localhost"，"test"，"root"，"").AsString )；
Conn.Open()；
```

如果你使用的是.net 的集成开发环境(visual studio.net)那么在代码中输入 MySQLDriverCS 后再输入一个”.”就可以看到 MySQLDriverCS 名称空间中的所有的东西了。

下面是 commond.

```
MySQLCommand cmd;
cmd = new MySQLDriverCS.MySQLCommand("DROP TABLE IF EXISTS
test.mysqldrivercs_test",conn);
cmd.ExecuteNonQuery();
cmd.Dispose();
cmd = new MySQLDriverCS.MySQLCommand("CREATE TABLE test.mysqldrivercs_test("+
"SettingID tinyint(3) unsigned NOT NULL auto_increment,"+
```

"SettingValue text, "+

"PRIMARY KEY (SettingID), UNIQUE KEY SettingID(SettingID), KEY SettingID_2 (SettingID))"+

" TYPE=MyISAM COMMENT='MySQL test table'",conn);

cmd.ExecuteNonQuery();

cmd.Dispose();

数据库的基本信息构建完成之后就可以实现对数据的录入。以下是数据库录入的关键代码。

string Value = "Value"；

int SettingID = 1；

new MySQLInsertCommand(conn,

new object[，] {{"SettingID"，SettingID}，{"SettingValue"，Value}}，

"mysqldrivercs_test"

)；

数据库不仅仅涉及数据的录入还涉及到数据删除, 下面所示的是 delete 部分的关键代码。

new MySQLDeleteCommand(conn，"mysqldrivercs_test"，new object[，] {{"SettingID"，"="，SettingID}}，null)；

关闭链接：

Conn.Close()；

5.3 URL 解析

private String _url = null；

private String _sysName = null；

private int _depth = 1；

public Html(String url，int depth)

{

This._url = url；

This._depth = depth；

}

public Html(String url，String sysName，int depth)

5.4 URL 队列管理

5.4.1 URL 消重处理

```
public class HtmlQueue ： IDisposable
{
private Queue<Html> _queue = null；
private SortedDictionary<String,  char> _visited = null；
public HtmlQueue(SortedDictionary<String,  char> visitedSites)
{
if (this._queue != null)
{
This._queue。Clear()；
This._queue = null；
```

5.4.2 URL 等待队列维护

对 URL 等待队列维护的具体操作就是进行对从队列中取出和增加 URLs 的维护。由于由于第一个网页所获取的等待 URLs 非常多，假如采取表格的形式对其进行存储，就会占据很大的内存，并且存取速度受到制约。因此，我们可以利用数据库实现对等待的 URLs 实现存储，然后建立两个缓存区实现对存取速度的提高的功能。其具体操作步骤如下：第一，当缓存存储按照规定达到一定的存储数目后，就可以利用 waitQueueDecrease()函数实现，把缓存中的底层数据（就是先存入的数据）放到数据库中进行等待。第二，当取出缓存中的数据被调用的一定程度后，就可以利用 waitQueueAdd()函数实现对数据库等待的 URLs 进行调用，使其存入取出缓存的对位。对于 MYSQL 数据库而言，每次调用数据都是以 50 条为以整体进行调用。

5.5 SOCKET Create()函数的设计与实现

SOCKET Create(int af, int type, int protocol) 函数主要是实现应用程序通过函数 socket 函数的调用，建立能够满足通信的网络套接字。我们可以看到该函数有三个参数，分为代表指定通信协议相关参数和对套接字类型的制定及程序所使

用的通信协议的指定。假如该函数在程序中调用成功就会返回一个被创建的套接字的整数类型的相关描述, 假如程序调用不成功, 就会返回一 INVALID_SOCKET。对每个进行而言，其存储空间中都存在一个套接字描述，该描述一一对应着存储的套接字数的结构，并且把所有的套接字描述建立成为一张表进行保存，从上面我们可以大致的判断出，该表是由两部分组成的，一部分专门放置套接字描述符，另一部分放置套接字相关的存储地址。下面为我们介绍 Create 函数声明，如表 5-1 所示。

表 5-1 Create()函数的声明及说明

函数声明	返回值类型	说明
SOCKET Create(int af,int type,int protocol	SOCKET	调用 Create 函数来创建一个能够进行网络通信的套接字

功能：使用前创建一个新的套接字

格式：SOCKET Create(int af, int type, int protocol)

参数：

af： 通信发生的区域

type： 要建立的套接字类型

procotol： 使用的特定协议

在应用 Create 创建 SOCKET 连接时要应用 WSAStartup( )函数和 WSACleanup( )函数来创建和验证版本号。

Create（）函数在论文中的具体调用图实现如图 5-1 所示：

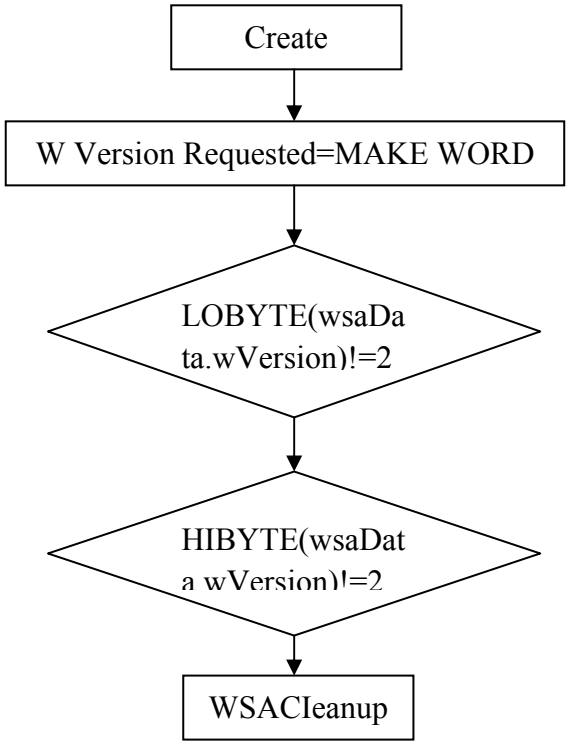

图 5-1 Create（）函数调用图

Socke 必须基于 WSAStartup 函数调用的基础上才能实现其使用。如上表所示 WSAStartup 函数有两部分组成，依次为利用高地位字节分别代表版本号的副、正来指明 Socket 版本型号的参数和获取版本号的相关信息。按照我们对 WSAStartup 函数的介绍，我们不难发现，在程序调用该函数时，系统需要做两个工作，第一个是对系统所装载的 Socket 版本进行读取和在 Socket 数据库中获取相关的版本信息，第二，把获取的版本号和相关信息一起返回个该函数。假如函数获取上述信息成功就会返回 0，一个软件系统一般只需要获取一次 Socket 版本号及描述信息，也就是只要获取到上述所有东西后，以后就可以随便调用 Socket 函数了。在完成上述获取版本号和相关信息后，我们最好调用 WSACleanup 函数实现对 Socket 库和其占有的相关资源的一起释放。

5.6CConnect( )函数的设计与实现

客户通过程序对 connect 函数的调用就可以实现，其与从事监听 name 工作的计算特定接口实现利用 Socket 进行相应的连接。如果连接成功，connect 返回 0；如果连接失败，则返回 SOCKET_ERROR。

int CConnect(SOCKET client, string dest)函数的主要是实现客户通过程序对 connect 函数的调用就可以实现，其与从事监听 name 工作的计算特定接口实现利用 Socket 进行相应的连接。该函数的函数表述如表 5-2 所示。

表 5-2 CConnect( )函数声明及说明

函数声明	返回值类型	说明
int Cconnect(SOCKET client,string dest)	int	客户程序调用 connect 函数使客户 Socket s 与监听于 name 所指定的特定端口上的服务 Socket 进行连接

功能：完成连接工作

格式：int CConnect(SOCKET client, string dest)

参数：

client：是由 socket()调用返回的并且未作连接的套接字描述符。

CConnect 函数接口说明如表 5-3 所示。

表 5-3 CConnect( )函数接口表

函数名	Cconnect()		
功能概要	连接到服务器		
记述形式	Int Cconnect(SOCKET client,string dest)		
参数			
类型	变量名	I/O	说明
SOCKET	Client	I	所创建的套接字
String	dest	O	
返回值	类型	int	说明
	值	-1	连接不成功,输出 connect error
		0	连接成功
详细说明	1、通过创建的套接字与服务器进行连接；		
	2、参数为创建的套接字的名字和连入服务器的名字。		
注意事项	所需要输入的服务器地址必须小于 50 字符。		

对应用实现中的说明如下：

在应用 connect 连接服务器与客户端时，要调用 sockaddr_in 地址解析函数

地址结构说明：

```
struct sockaddr_in
{
short sin_family；              //AF_INET
u_short sin_port；              //16 位端口号，网络字节顺序
struct in_addr sin_addr；       //32 位 IP 地址，网络字节顺序
char sin_zero[8]；              //保留
}
```

定义参数 result；

result = connect(client, (SOCKADDR*)&addrSrv, sizeof(SOCKADDR))；

说明：

client：所创建的 socket 连接。

(SOCKADDR*)&addrSrv：主机的地址。

sizeof(SOCKADDR)：地址的长度。

然后对参数 result 是否成功建立判断，假如成功建立则返回值为 0，并且利用 WSACleanup（）函数结束 Socket 的相关调用，并释放相应的系统资源。

5.7 CClose( )函数模块的设计与实现

void CClose(SOCKET client)用来关闭套接字。CClose( )函数具体表述如表 5-4 所示。

表 5-4CClose( )函数声明及说明表

函数声明	返回值类型	说明
void Cclose(SOCKET clinent)	void	客户程序调用 Ccloset 函数来关闭 Socket 进行连接

功能：关闭套接字 client

格式：void CClose(SOCKET client)；

CClose 函数在论文中的应用如下：

```
void Csocket：：CClose(SOCKET client)
{
closesocket(client)；
WSACleanup( )；
}
```

说明：

调用 closesocket()函数关闭 socket 连接；调用 WSACleanup( )函数释放 socket 所占用的资源。

CClose( )函数接口具体表述如表 5-5 所示。

表 5-5 CClose( )函数接口表

函数名	CClose()；		
功能概要	关闭套接字		
记述形式	void Cclose(Socket Client)；		
参数			
类型	变量名	I/O	说明
SOCKET	Client	I	所创建的套接字
返回值	类型	int	说明
	值	0	成功关闭套接字 client
详细说明	关闭创建的套接字		

5.8 CSend( )函数模块的设计与实现

CSend(SOCKET client, const char* sendbuf, int len) 函数的主要是实现客户程序用 send 函数向服务器发送请求。CSend( )函数具体表述如表 5-6 所示。

表 5-6 CSend( )函数声明及说明表

函数声明	返回值类型	说明
intCsend(SOCKET client,constchar*sendbuf,int len)	int	客户程序调用 send 函数向服务器发送请求

说明：

功能：数据的发送

格式：int CSend(SOCKET client, const char* sendbuf, int len)；

参数：

Sendbuf：指向存有传输数据的缓冲区的指针；

len：发送数据的长度。

CSend( )函数的接口具体表述如表 5-7 所示。

表 5-7 CSend( )函数接口表

函数名	Csend()		
功能概要	发送数据到服务器		
记述形式	Int Csend(SOCKET client,const char*sendbuf,int len)		
参数			
类型	变量名	I/O	说明
SOCKET	Client	I	所创建的套接字
const*char	sendbuf	O	保存 http 的请求包
int	len	O	所发送的数据长度
返回值	类型	int	说明
	值	0	发送的字符数
详细说明	向服务器发送由用户输入的目标地址的数据		

CSend 函数在本论文中的实现如下所示：

```
int Csocket：：CSend(SOCKET client,  const char* sendbuf,  int len)
{
int result；
result = send (client ,  sendbuf ,  len ,  0)；
return result；
}
```

说明：

CSend(SOCKET client, const char* sendbuf, int len)向服务器发送数据请求。设立参数 result，返回发送请求。

5.9 CRecv( )函数模块的设计与实现

void CRecv(SOCKET client, FILE* fp)函数的主要是实现接受服务器所返回的数据信息。函数 CRecv( ) 具体表述如表 5-8 所示。

表 5-8 CRecv()函数声明及说明表

函数声明	返回值类型	说明
voidCRecv(SOCKET Client,FILE*fp)	void	客户应用程序都用 Crecv 函数从 TCP 连接的另一端接收数据

说明：

功能：数据的接受

格式：void CRecv(SOCKET client, FILE* fp)

参数：

Client：所创建的 socket 套接字链接；

Fp：创建用来保存目标 URL 的文本文件。

函数 CRecv( )接口具体表述如表 5-9 所示：

表 5-9 CRecv()函数接口表

函数名	Crecv()		
功能概要	接收由服务器发送过来的数据		
记述形式	void Crecv(SOCKET client,FILE*fp)		
参数			
类型	变量名	I/O	说明
SOCKET	Client	I	客户端所创建的套接字
FILE*	fp	O	保存接收到的目标 URL
返回值	类型	void	说明
	值	-1	没有接收到数据或者数据不符合要求
		0	如果数据正常则可以正常过滤并保存
详细说明	fp 为目标文件名		

5.10 主要类的实现

URL 是 CDcURL 类的一个我们非常熟悉的实例代表，对 URL 的解析主要通过对通 ParseURL 函数的调用和对 URL 的字符串进行传递。然后把获取其主机名，IP 地址等等私有变量成员保存起来。然后利用 GetHostName 函数和 GetHostIP 等公共函数实现对 class CDcURL 中上述信息的获取。具体操作的关键代码如下：

```
{
public:
CDcURL(void);
~CDcURL(void);
public:
string                          m_sURL;
string                          m_sHostName;
string                          m_sIP;
string                          m_sRequestPath;
unsigned int                    m_nPort;
//static var
static int                      m_nObjectCnt;
static bool                     m_bInited;
static list<string>             m_listUnVURL;
static list<string>             m_listVURL;
static list<string>             m_listNeedVURL;
};
```

CDcURL中存在着许多静态变量，主要是为了保存已经或者还没访问过的URL信息，达到保存所有URL实例的功能。通过类的成员函数对静态变量的调取，具体操作的关键代码如下：

```
static list<string>             m_listUnVURL;
static list<string>             m_listVURL;
static list<string>             m_listNeedVURL;
bool CDcURL::AddOneUnVURL( string &unvURL )
{
list<string>::iterator i;
```

```
for( i = m_listVURL.begin(); i!=m_listVURL.end(); i++ )
{
if( unvURL==*i )
return false;
}
for( i = m_listUnVURL.begin(); i!=m_listUnVURL.end(); i++ )
{
if( unvURL==*i )
return false;
}
m_listUnVURL.push_back( unvURL );
return true;
}
bool CDcURL::AddOneVURL( string &vURL )
```

爬虫工作部分具体操作的关键代码如下：

```
class CDcSpider
{
public:
CDcSpider(void);
~CDcSpider(void);
public:
bool                    Start( int nThreadCnt, string saveFilePath );
void                    Wait();
bool                    End();
public:
static bool                 s_bContinue;
static CRITICAL_SECTION s_csURLSection;
static DWORD WINAPI     OneSpiderThread( LPVOID lpParameter );
private:
HANDLE                  m_hThread[10];
int                     m_nThreadCnt;
};
```

Start虽然启动了爬虫，但是对启动的爬虫的个数和保存位置没有具体的设定。但是通过下述程序就能实现Star对爬虫数量进行控制，其具体是通过控制爬虫线程来实现的。与Start功能相反，End让各个爬虫线程结束工作，Wait让各个爬虫线程等待线程的结束。由此可见，End建立在Wait的。具体操作的关键代码如下：

```
bool CDcSpider::Start( int nThreadCnt, string saveFilePath )
{
if( CDcSpider::s_bContinue )
return false;
if( nThreadCnt>10 || nThreadCnt<0 )
return false;
m_nThreadCnt = nThreadCnt;
CDcSpider::s_bContinue = true;
InitializeCriticalSection( &CDcSpider::s_csURLSection );
```

利用网络爬虫爬获取的网页相关信息只有经过一些处理，等到能够可以利用的URL后，才能间接的存贮到本地系统下的。然后把获取的URL相关信息保存在相关的列表中，以方便后面进行调用。上述的对页面进行处理是通过CDcPage类来实现的。其具体操作的关键代码如下：

```
class CDcPage
{
public:
string              m_sRawPageContent;
string              m_sFinalPageContent;
string              m_sPageHeader;
CDcURL              m_dcURL;
CDCSocket           m_dcClientSocket;
vector<string>m_vecFindUrl;
vector<string>m_vecInvalidUrl;
};
```

ParserCurPageURL()函数主要是为了实现通过爬虫线程多的网页数据的香瓜解析，一伙的可以利用的 URL 信息的。然后把获得的信息添加到还未访问的 URL 列表中。其具体操作的关键代码如下：

```
int CDcPage::ParseCurPageURL()
```

```
{
firstPos = m_sFinalPageContent.find( "<a ", secondPos );
if( string::npos==firstPos )
break;
secondPos = m_sFinalPageContent.find( ">", firstPos );
blankPos = m_sFinalPageContent.find( " ", firstPos );
if( string::npos!=blankPos )
{
if( blankPos<secondPos )
secondPos = blankPos;
}
```

5.11 本章小结

本章节通过对实现爬虫线程操作的系统过程利用的各种工具的具体介绍，然后对系统所能实现的相关的客户界面进行了介绍。并对系统一些主要实现上述工程的具体函数和细节进行了总结和详细介绍。

第 6 章 系统测试

6.1 系统性能测试理论基础

虽然，软件测试是系统开发过程中的最后一个阶段，但是他的重要性已经得到了同行所有人的认可。软件测试不仅仅是对软件是否满足客户需求、能达到需求计划中要求的各种功能的正常实现。还是检查软件在各个运行环境中能否正常运行的保证和支撑，更是测试各个界面和编译的功能能否让用于实现简单和快速的操作。可以说软件测试是软件能否实用化的关键之一。

软件测试一般包括测试功能是不是健全、性能是不是稳定、对信息和数据处理设计的服务器是否能到满足大访问量的需求、用户界面设计是否合理等等。由于现在的软件都是集成了许多开发工具于一体进行开发而得到的，比如数据库、编程软件等，并且他们还有和相应的硬件和网络通信相适应，才能实现其功能。因此，现在的软件测试越来越复杂和经历的时间越来越长，在软件开发中所占的时间和资金投入比例都越来越大。对于一个已经设计好的新系统，它的软件测试中的性能测试具体步骤如下：

第一、测试的文件准备和人员组织阶段。在性能测试前期，我们要根据客户的需求计划中所设计的所有性能进行整体和归纳，实现对测试目标的确定，然后按照目标，制定如何进行测试和对测试人员按照测试目标进行分配。

第二、测试过程和环境的设计。性能测试就是要按照不同客户对软件操作的情况进行测试。由于人和人之间存在知识差异的同时也存在文化差异，并且每个人都有一些自己的习惯动作。第二阶段就是要设计出涵盖几乎所有人对软件操作中可能出现的问题或者不合理的操作方式的归纳和整理，然后按照上述整理主义进行测试。

第三、对测试结果进行分析。假如说第二阶段是测试的主要阶段，那第三阶段就是测试的关键和最难的阶段。由于每个编程人员的编程习惯和各个编程人员的合作及总体需求的设计等等 各个方面难免出现漏洞。所以，这对测试结果分析就带来了不小的考验，测试员按照测试结果各处的各个数据，让结合上述种种因素的基础上，凭借经验和知识双重保证的基础上，才能完成第三个阶段的任务。

上面我们对性能测试的三大步骤进行了总结，接下来我们总结性能测试的主要目的：第一，模拟真实的用户操作水平和操作习惯，在模拟现实软件使用时，可能遇到的各种用户的操作习惯和知识水平，对他们进行归纳和整理，选择出具有代表性的、能够充分体现真实的客户操作进行模拟。第二，对系统服务器的压力测试，主要通过大量的客户同一时间对系统进行访问，然后检测系统能否满足客服的需求，能在规定的预计的单位时间的访问人数上达到要求。通过对以上两种主要目的的实现，就可以检查系统性能是不是能够达到标准。

现阶段，我们对系统测试主要是建立在以下几个进行测试基础之上的。

1、单元能力测试，测试员对所有的系统中包含的单元都要在测试指出制定详细的具有针对性的测试方式和测试手段，必须具体到测试所有的代码。但愿能力测试是系统测试的基础，是保证系统各个部件能够独立运行的保证，更是系统能够整体高效运行的基石。如果，单元能力测试都达不到标准或者需求计划的要求，那整体系统就很难达到客户满意的程度。

2、低负载能力测试，低负载能力测试主要是从系统整体角度对系统包含的所有单元进行综合测试的手段，看系统能否在低负载的时候满足设计阶段制定的目标。假如系统在低负载时都不能达到设计要求，那在比较高的负载下一定会达不到客户的要求，因此，低负载是否能够达标，是进行高负载测试的基础。

3、利用真实化的测试数据，性能测试是否能够真实地可靠的模拟现实生活中系统的使用状况和可能出现的问题。这就要看在系统测试中能否提供一个能够模拟现实中系统所使用的环境能否一致、操作人员的技术和能力是否能够模拟到位的比较真实地、全面的详细的数据作为测试的支持。假如没有真实化的数据作为支持，那性能测试的结果就不能十分有把握的保证系统能够满足实际的需求。

6.2 详细测试过程

在详细测试过程中，本文作者主要对网络爬虫程序在防火墙模式下和覆盖率模式下对系统的所有性能和各种模式进行了测试。下面具体说明这两种模式下，系统性能的表现。

网络爬虫程序在防火墙模式下虽然通信能力和速度都会加强，同时减少通信的资源消耗，但是会影响系统的覆盖率和整体的质量。在本章节中，本文作者将通过在防火墙模式喜爱对网络爬虫程序进行定量的针对其效率存在的问题，通过多种试验，总结数据进行分析。在试验中，本文作者选用了多线程网络爬虫模式

在并行方式下运行效率在防火墙模式下的覆盖率的体现。

实验的准备阶段所做的具体操作。通过对四千万网络页面相关数据的收集，作为原始的测试数据。利用哈希分块算法，通过一定方式随机的从这四千万个数据中抽取5个数据，然后进过相关处理作为URL的初始数据。总理论上讲，假如有五的倍数的URL初始数据被使用，则网络爬虫进程数应该是11或者其倍数。这是由于网络爬虫在防火墙模式下进行访问页面时，只能获取页面的内连接，而不能进行其外连接。在防火墙模式下，我们在广度优先原则的基础上，通过对所有网页的URL通过网络爬虫程序进行了获取，并在其全部下载完成后。通过对相关数据的归纳、整理和总结分析，对网络爬虫的整体覆盖情况进行了研究。在获取数据方面，我们按照初始的设置，分别采用五的倍数进行初始YRL，并且初始所使用的URL在每个测试中都不一样。但是，经过对所有获得的试验数据进行总结，我们发现每次的试验结果都非常相似，除去误差，可以理解为相同。

试验数据的汇总如图6-1所示，其横坐标代表并行方式喜爱网络爬虫的数量，纵坐标代表其获得网页页面的整体覆盖率。图中那个曲线就是并行方式下我们对网络爬虫测试的结果。从试验结果及曲线我们可以看出，在网络爬虫只有一个的时候，其覆盖率仅仅只有百分之九十。造成这样结果产生，我分析主要是因为我们在使用一个网络爬虫进程时只有五个URL参与了初始化而我们理论分析时期初始化数据都在一千万左右。按照这样计算，我们这个实验由于初始化的URL比较少，就不能对四千万个网络页面进行全部的抓取或者使用。我们从这个数据统计图表中，还可以发现，在防火墙模式下，网络爬虫进程和其覆盖率成反比。这是因为，随着网络进程的不断增加，网页就会被逐渐细分为很多更小的区域，这些区域的产生，增加了外连接模块。这就使得在防火墙模式下，有很多网页不能被访问。从图表中曲线的走势我们可以看出，刚开始的时候其斜率不是很大，我们仔细观察发现，斜率的第一个拐点是4，第二个拐点是64.。具体的说在网络爬虫的总体进程数小于四时，其覆盖率随着网络进程的增加，其减小的速度不是很快。

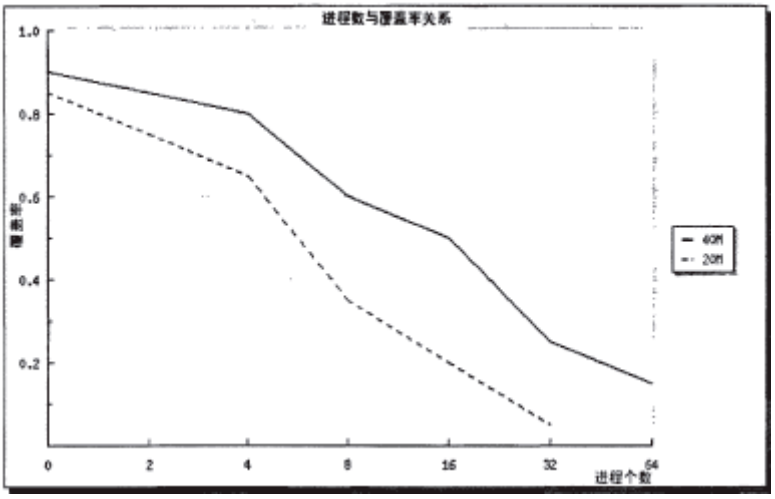

图6-1 进程数与覆盖率关系

从上述的实验结果，我们不难想到，在防火墙模式下，覆盖率和网络爬虫进程中的URL初始的个数存在很大的联系。随后，我们采取不同数目的URL初始个数，在防火墙模式下，对该实验就进行了多次测试。通过对测试结果的总结，我们的测试结果总结为如图6-2所示。横坐标表示网络爬虫程序的URL所有初始化的个数，纵坐标表示其覆盖率。

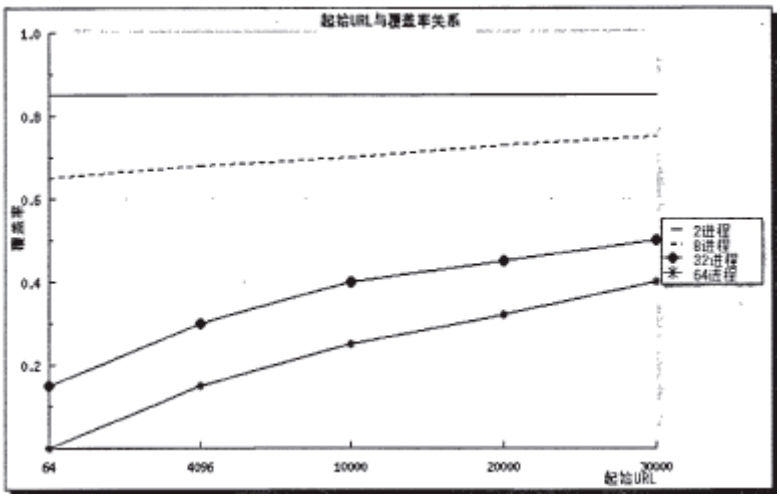

图6-2 起始URL于覆盖率关系

通过对上述图表的观察，和相关数据的总计和分析，我们可以得到下面几点结论：

第一、在并行方式下的网络爬虫程序使用的进程数比较少时，可以在防火墙模式下使用的网络爬虫程序获得比较好的覆盖率，使用效果比较好。在此时，网

络进程需要的URL初始值的个数不是很多，这就减少了URL初始值个数对覆盖率的影响程度。

第二、要实现对网页中所有的子网页实现所有覆盖和访问，选择防火墙模式下的网络爬虫程序进程访问，不能达到理想的要多。因为在防火墙模式下，网络爬虫进程和其覆盖率成反比。这是因为，随着网络进程的不断增加，网页就会被逐渐细分为很多更小的区域，这些区域的产生，增加了外连接模块。这就使得在防火墙模式下，有很多网页不能被访问。

以上结论都是基于在四千万网页数据的基础上经过多次试验和不断分析得出的结论。这就会让我们产生疑惑，是不是这些数据选择不是很合适，恰好选择在了网络爬虫程序不是很擅长的地方，是不是换其他的数据就能实现或则得到理想的结果呢。还有是不是我们不选择四千万个数据，而是采用上亿个数据进行测试，是不是数据的多少对网络爬虫在防火墙模式下的覆盖率有影响呢。对于这些问题的判断，我们很难做出比较让人满意的回答。其原因如下：假如我们新增加的数据和原有的数据存在着一定的联系或者关联，这就在表象上产生在防火墙模式下，覆盖率提高的非常快。第二，假如这些新增的数据和原有的数据不存在任何关联度或者联系，我们就可以发现，在防火墙模式下，其覆盖率降低的幅度非常大。因此，要解决增加数据对覆盖率是否产生影响，就得看增加的数据和原有的数据之间是否存在一定的联系或者关联度比较高。

上述我们对网络爬虫程序在防火墙模式下对网络页面覆盖率影响问题研究中。我们选取了两百万个网页参与了测试，这些网页的选取是随机的，几乎覆盖了所有网页的一半还要多。在获取这些网页的相关数据后，我们在数据中随机的抽取了一半数据参与到试验中，并且这些数据之间不存在关联，通过实验和数据整理，我们绘制出了图，图6-1中所示的点状图。从对这些数据的相关分析我们可以看出，假如在网络页面不断增加的同时，还要保证在防火墙模式下还具有相同的覆盖率，就可以通过网络爬虫的进程个数的相应增加来达到上述结果。换个方式说就是，新增加的许多网页的相关数据，可以通过相应增加的网络爬虫进程的个数来实现访问，从而保证覆盖率没有影响。

6.3 研究成果

结合以上的测试，我们将本次课题所研究的网络爬虫系统以图片的方式进行了一次成果展示。

首先，在网络爬虫系统中输入需要遍历的URL或者IP地址。如图6-3所示。

图6-3 输入URL

当确认参数输入正确后，点击“添加任务”按钮，如果程序检查正确，域名将被加入到左边的待遍历列表中。如图6-4所示。

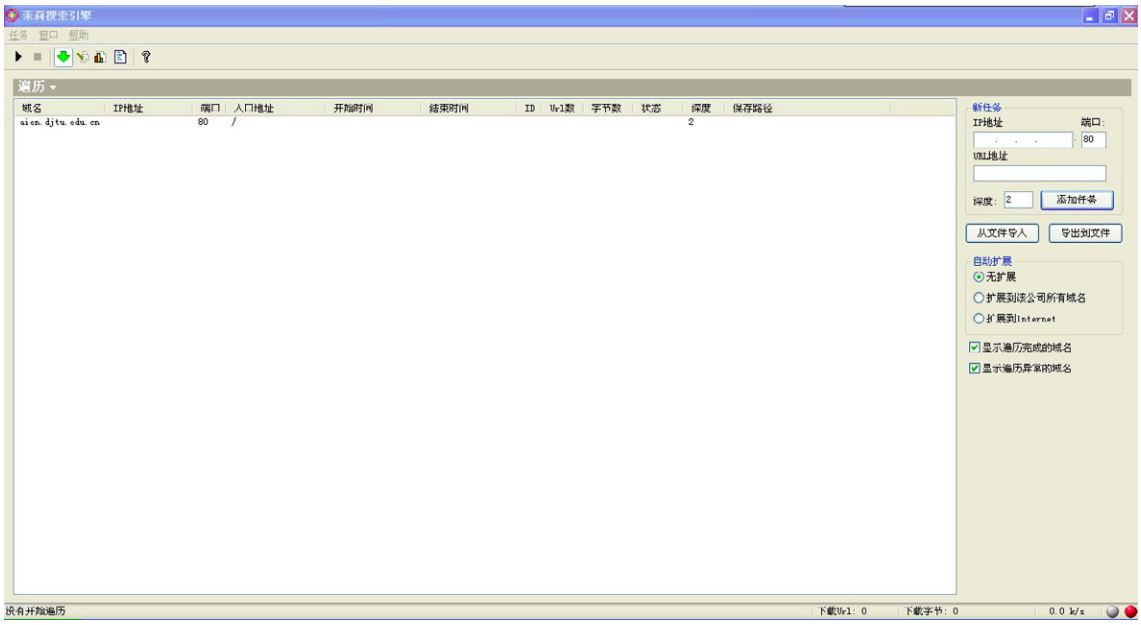

图6-4 输入URL之后的操作界面

接着点击菜单栏中的“启动”选项启动遍历任务。如图6-5所示。

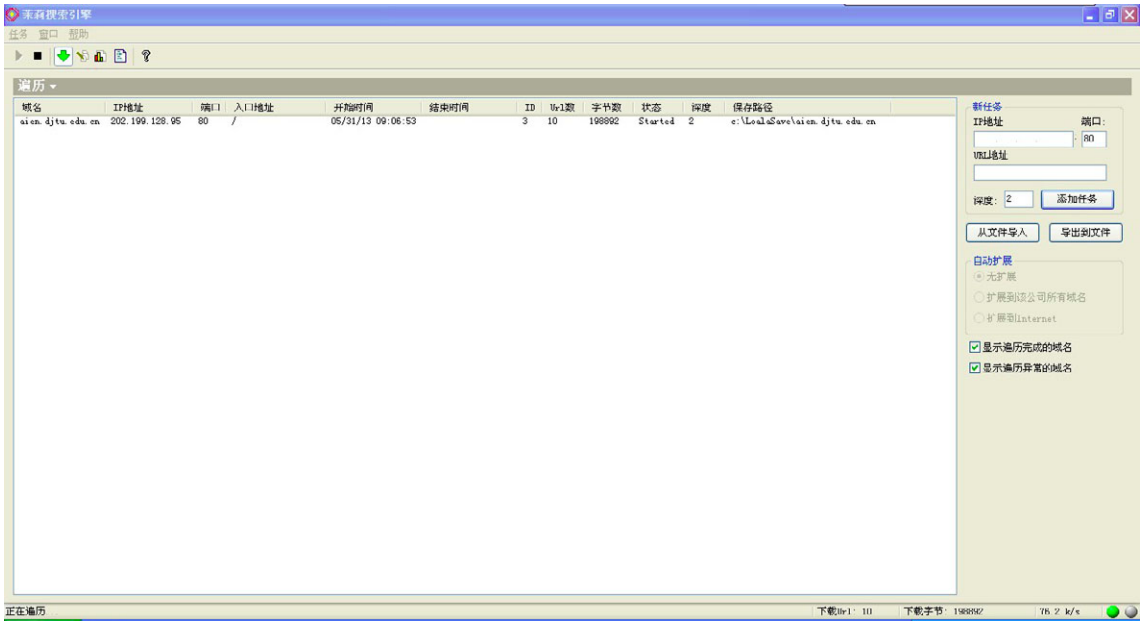

图6-5 点击“启动”选项

当遍历域名已经达到最大的深度后，遍历任务结束。如图6-6所示。

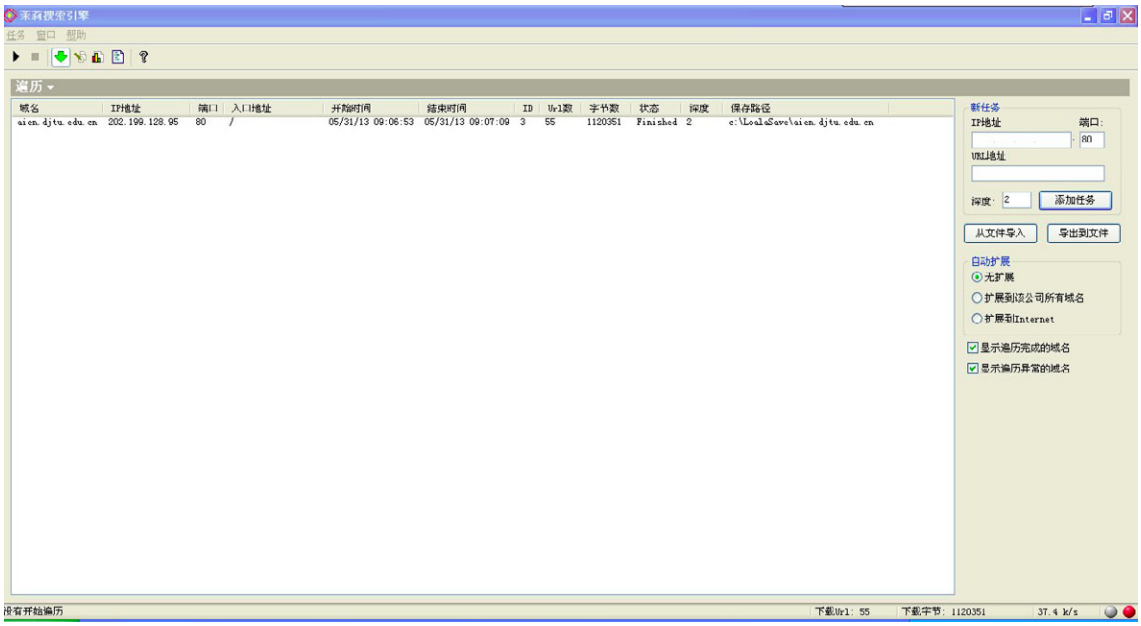

图6-6 遍历结束

在遍历列表中查看实时的统计信息。如图6-7所示。

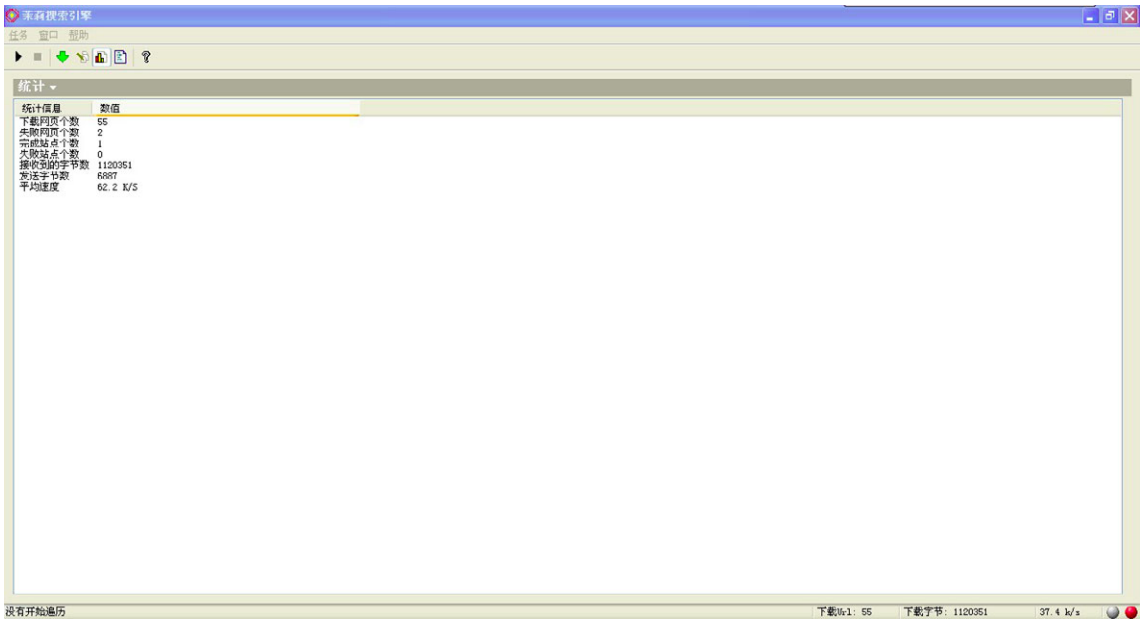

图6-7 实时统计信息的查看

在“日志”页面中显示实时日志。如图6-8所示。

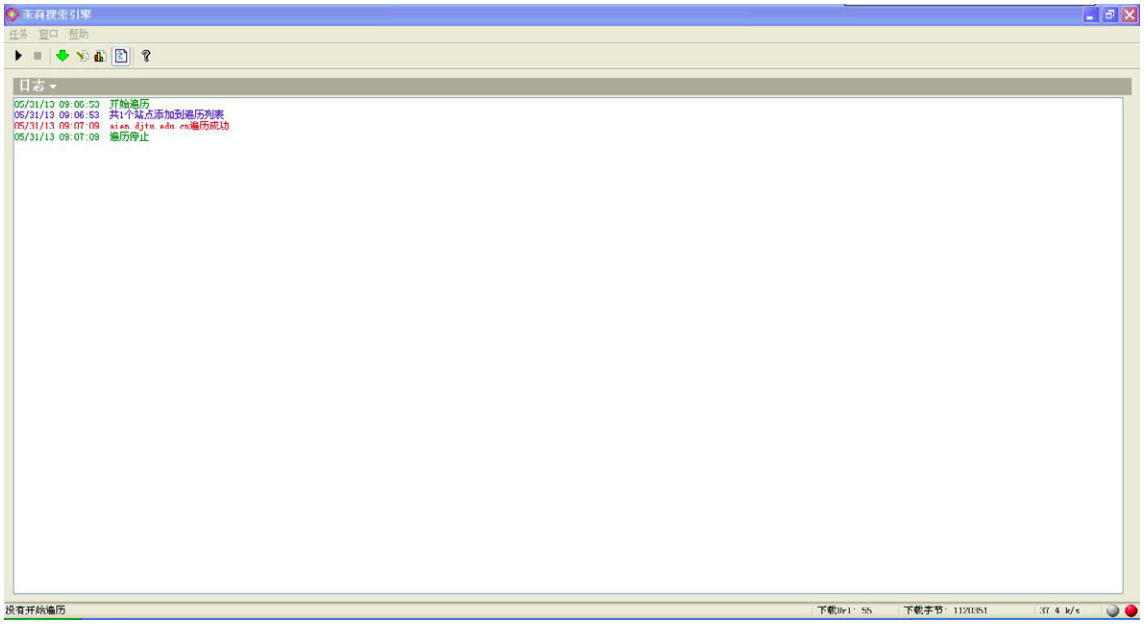

图6-8 实时日志显示界面

抓取下来的页面保存在本地。如图6-9所示。

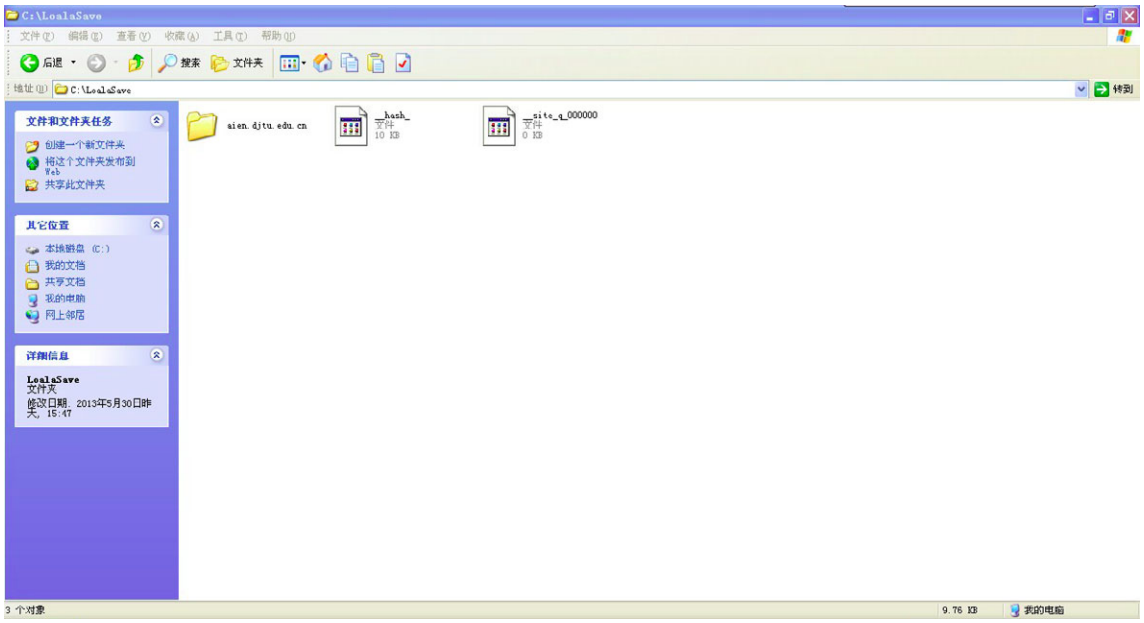

图6-9 在本地保存的抓取的页面

下图显示的为本地保存的抓取的页面数量和目录。如图6-10所示。

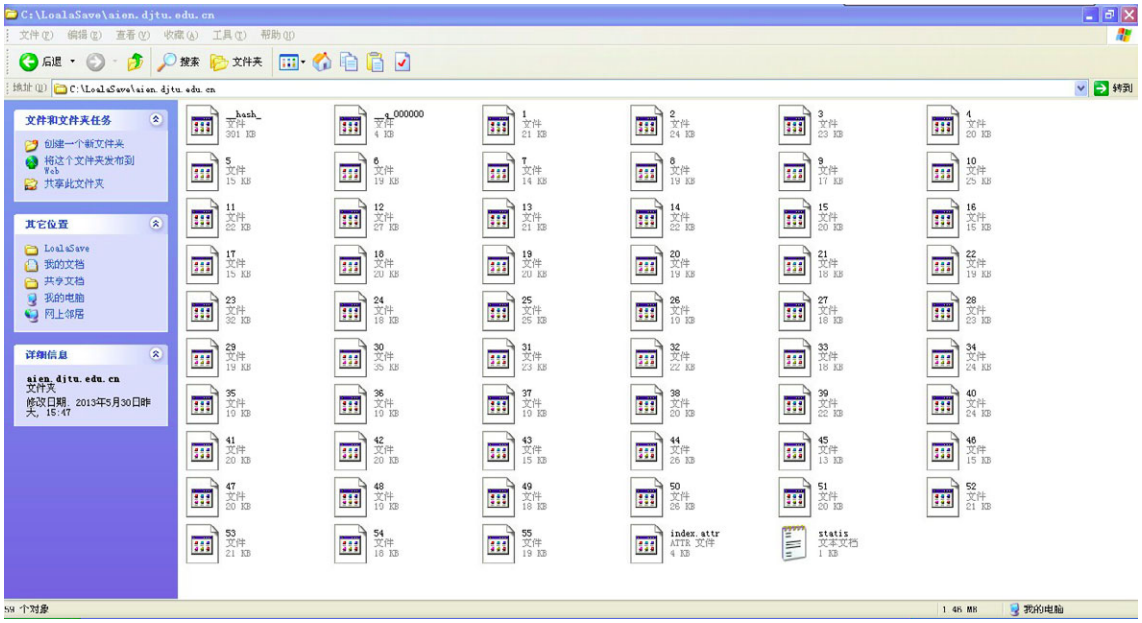

图6-10 抓取页面的显示

第 7 章 结论

网络爬虫程序通常被用来为搜索引擎、数据缓冲，数据挖掘等服务提供数据来源。随着互联网体积的不断增大，把单进程的网络爬虫架构更改为并行网络爬虫就显得非常重要了。遗憾的是在公开化的文献中几乎没有文章对并行网络爬虫技术的性能进行过对比测试。本文正是由于这个原因对各种文献中介绍的各种网络爬虫技术进行汇总对比，并研究它们各自的性能。当然，文中的实验数据能够为网络爬虫程序的设计者提供有益的指导。

最后，再总结一下文中的研究成果：

当网络爬虫程序中只有少量的进程存在时(4个或更少)，使用防火墙模式会带来好的覆盖率。因为防火墙模式的网络爬虫程序完全独立运行并且实现起来相对简单，所以，这将是一个不错的选择。下面两种情况中，防火墙模式的网络爬虫程序就不适合了：

1、需要多于4个网络爬虫进程进行工作。

2、当只需要下载网络页面的很少一部分，也就是说网络爬虫程序下载网络页面的质量显得非常重要。

基于交换模式的网络爬虫程序由于URL的交换会消耗很少量的网络带宽(少于1%), 同时, 还可以采用批处理信息交换技术使其他的系统开销达到最小化的程度。在实验中，即使网络爬虫程序在工作中对回指链接交换的次数少于100次时，依然可以使网络爬虫程序下载的网络页面质量达到最大化的程度。

通过在各个网络爬虫进程间复制10000到100000个常用链接的技术，可以使网络爬虫进程间通讯开销减少大概40%。同时，如果复制更多的常用链接，并不能更有效的减少网络爬虫进程间通讯带来的开销。

本文进行了网络爬虫系统的设计与开发。在实践经验基础上，设计并开发了使用网络爬虫程序的内容呈现页面。取得的主要成果有以下几点：

1、对网络爬虫程序的整体流程和使用的技术有一定的研究。对网络爬虫并行化处理、各种拓扑结构做了比较详细的研究；

2、对网络爬虫技术涉及到的评测指标做了比较详细的研究。对于如何在各种拓扑结构下设置网络爬虫程序以及网络爬虫程序在各种拓扑结构下的表现方面做了一定的研究；

3、在实际项目中，如何选择最适合的网络爬虫程序等做了一次尝试与分析；

4、完成了对网络爬虫程序下载网络页面内容呈现的网站代码，并将其作为Module的形式可以使用drupal信息发布系统运行。可以说，在这次设计和开发过程中，学到了不少东西，尤其是对软件工程以及网络编程相关技术做了一个深入的探索。

当然，由于时间的原因和项目进度的安排，此次研究仍在进行之中，所以还有一些方面需要改进。归纳一下，主要是以下几点：

1、使网络爬虫程序可以采用分词(包括中文分词)的方法对网页内容进行进一步数据挖掘和采集工作；

2、使页面呈现程序可以展示更多的信息；

3、提高性能；

4、提高复用性，尽可能支持多种信息类型的信息获取。

在继续开发过程中，也一定会对这些技术与知识有一个更为深刻的认识。

致 谢

在本次毕业设计即将完成之际，首先要对我的导师进行诚挚地感谢，导师不但在论文的书写上对我给出了巨大帮助，而且从生活上对我进行了无微不至的关心与爱护。特别是导师对该论文从选题、构思、资料收集到最后定稿的各个环节给予细心指引与教导，使我得以最终完成毕业设计。老师严谨的治学态度、丰富渊博的知识、平易近人的处事作风、精益求精的工作态度、积极进取的科研精神以及诲人不倦的师者风范是我终生学习的楷模。另外，在读研学习期间老师鼓励我为勇于实现自己的理想而不断进取。在此向导师致以最诚挚的感谢和最衷心的祝愿。

其次，在学习的过程中，一直以来在不断得到各位老师和同学的关心与帮助，使我克服了无数来自生活与学习的困难，不断地完善自己，

我在学习和生活中不断得到友谊的温暖与关怀，最重要的是一种精神上的激励，让我非常感动。另外，论文初稿，课题实现研究方法、技术路线、实验方案也得到了许多同学的宝贵建议，同时还得到许多在工作过程中许多同事的支持和帮助，在此我要特别感谢对我进行过帮助的各位老师以及和我共同学习的各位同学、与我共同工作的同事，谢谢你们！

再次，我要对我的家人进行感谢，一直以来，他们对我进行无尽的关心、鼓舞、支持，从而使得我能够专心致志的投入到学习和研究工作中，以此顺利地完成学业。

最后，我要向在百忙中抽出时间对本文进行审阅、评议并提出宝贵意见的各位专家表示衷心地感谢！

谢谢！

参考文献

[1].刘世涛等, 简析搜索引擎中网络爬虫的搜索策略[N], 阜阳师范学院学报(自然科学版), 2006。60～63。

[2].吴小竹等。基于 C#的多线程 SPIDER 的设计与实现[J]。福建电脑，2004。62～63。

[3].胡宏涛等。基于网络的信息获取技术浅析[J]，福建电脑，2006。60～61。

[4].李学勇等。网络蜘蛛搜索策略比较研究[A]，计算机工程与应用，2004。131。

[5].高克宁等。支持 Web 信息分类的高性能蜘蛛程序[J]，小型微型计算机系统，2006。1309～1312。

[6].吴强等。C#线程[A]，C#SE 平台程序设计和实战，2004。103～135。

[7].荣传湘等。搜索引擎中数据获取的设计与实现[J] 沈阳：小型微型计算机系统，1999。

[8].Michelangelo Diligenti。 Frans Coetzee， Steve Lawrence， etal。 Focused Craw ling using Context Graph s[J]。 Intemat ional Conference on Very Large Databases。 2002 ： 527～534。

[9].叶允明等。 分布式 Web Crawler 的研究： 结构、算法和策略[J]。 电子学报， 2002。12 。

[10].David S。 Doermann， Ehud Rivlin and Isaac Weiss， Logo Recognition Using Geometric Invariants。 Document Analysis and Recognition。 Proceedings of the Second International Conference on， 20-22 1993： 4~34

[11].Jie Zou， Nagy G。。 Evaluation of model-based interactive flower recognition，Pattern Recognition。 2004。 ICPR 2004。 Proceedings of the 17th International Conference on, Volume 2， 23-26 Aug。 2004 Page(s)： 311~314

[12].Abo-Zaid。 A simple invariant neural network for 2-D image recognition。 Thirteenth National Radio Science Conference， 1996。 NRSC '96 ： 18~21

[13].Junhong Li， Quan Pan，Hongcai Zhang， Peiling Cui。 Image recognition using Radon transform。 Intelligent Transportation Systems， 2003。 Proceedings。 2003 IEEE： 995~1000

[14].Tuan D。 Pham。 Recognition of Trademarks with Spatial Statistics and Neural Learning。 Proceedings of the 2002 IEEE International Conference on Artificial Intelligence Systems ， 2002： 89~144

[15].Fang G Nicholascr。 Collaborative Project Management Architecture。 Proceeding of the 36th Hawaii International Conference on System Sciences，2002： 234~240

[16] 黄维晋，盛浩。JSP网站架构与实例。北京：大恒电子出版社，2002
[17] 路秋丽，余胜泉。面向学习对象的网络课程设计与开发。中国电化教育，2005，(1)：75-79
[18] 韩晓玲，李志文，段峰。基于Internet的网络教学环境。现代教育研究，2002，(11)：48-50
[19] 胜泉，何克抗。网络教学平台的体系结构与功能。中国电化教育，2001，(8)：60-63
[20] 刘万锁等。利用WEB数据库开发基于Internet的远程教学系统。中国远程教育，2000，(12)：48-50
[21] 张立君，蔡小秧。基于Internet的网络教学平台自学与自测系统设计。北京印刷学院学报，2005，13(4)：22-25
[22] 刘晶。网络课堂教学平台：[硕士学位论文]。天津：天津工业大学，2003
[23] 魏娟丽。基于ASP的网络考试系统的研究与开发。计算机与网络， 2004，(21)：59-60
[24] Hopper。S：Cooperative Learning and Computer—based Insturetion。 Educat ion Technology Research&Developmen，1992
[25] JoelMillecan等著。NTERNET信息服务器技术。北京：清华大学出版社，2004
[26] 易谅容，陈志刚。网上教务管理系统的开发与实现。系统工程，2002，20(6)：87-90
[27] 刘志明。基于工作流技术的项目管理系统设计与实现：[硕士毕业论文]。湖南：湖南大学，2009
[28] Wil van der Aalst，Kees van Hee。工作流管理：模型、方法和系统。北京：清华大学出版社，2004
[29] JayGreenspan，BradBulger。MYSLQ/PHP数据库应用开发指南。北京：电子工业出版社，2001
[30] Aaron Saray。PHP设计模式。北京：清华大学出版社，2010
[31] 宋敬彬，孙海滨等。Linux网络编程。北京：清华大学出版社，2010
[32] 曾强聪，赵歆。软件工程原理与应用。北京：清华大学出版社，2011

专业学位硕士学位论文

MASTER THESIS FOR PROFESSIONAL DEGREE