基于 Python 的多线程网络爬虫的设计与实现

◆孙　冰

（中国石油大学计算机与通信工程学院　山东　266580）

摘要：本文主要详细介绍如何应用 Python 语言实现一个多线程的网络爬虫程序，并在此基础上搭建特定的测试网站将串行爬虫程序和多线程爬虫程序的运行效率进行对比，进而给出提高网络爬虫性能的具体方法。

关键词：Python；网络爬虫；多线程

0　引言

随着网络技术的飞速发展，互联网中的信息呈现爆炸式的增长，互联网的信息容量也达到了一个前所未有的高度。为了方便人们获取互联网中的信息，国内外出现了一批搜索引擎，如 Google、百度、Yahoo 等等。这些搜索引擎的特点是能尽量多地抓取网页中的信息，因而容易忽略抓取到的页面的语义和抓取到的顺序等。检索人需要投入大量时间和精力来完成一次检索，必要时还需要反复组织自己的检索语言，以达到检索的效果。传统的搜索引擎在返回的结果方面有局限性，网络爬虫因此而诞生。网络爬虫又名叫网络机器人，它是一种按照特定规则爬取网页信息的程序。与传统搜索引擎不同，网络爬虫只爬取想要获得的特定类型的信息，进而提高搜索引擎的效率。

传统的搜索引擎通常由网页搜集、预处理和查询这三个模块组成，而网络爬虫就存在于网页搜集这个模块之中，网络爬虫作为搜索引擎[1]的重要组件，它的主要功能就是爬取互联网上各类信息。网络爬虫通常是一个应用程序或者脚本，一般先给定一个入口 URL 地址，从入口 URL 开始根据一定的规则获得这个初始网页上的所有 URL，再通过这些新的 URL 如此循环往复获得更多的 URL。在这些获取到的 URL 中，按照我们需要信息的规则解析该网页，最后再根据不同的需求对获取到的数据进行处理。网络爬虫与传统检索方式对比如图 1 所示。

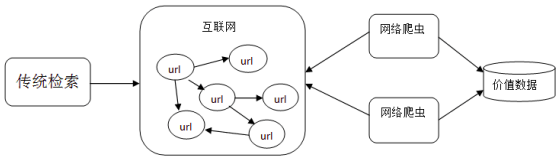

图 1 网络爬虫与传统检索方式对比图

理论上任何一种支持网络通信的语言都可以用来编写爬虫程序，目前大多数网络爬虫程序是用后台脚本语言来编写，其中 Python 是最为广泛使用的一种语言，其具有丰富和强大的标准库供用户选择使用[2]。

1　网络爬虫的设计实现

本文采用简单的框架结构来编写爬虫程序，分别有以下四个模块[3]：URL 管理器、网页下载器、网页解析器、网页输出器，这四个模块共同完成抓取网页的整个过程。

（1）URL 管理器

URL 管理器模块的作用是管理待爬取的 URL 集合和已爬取过的 URL 集合。每个网页爬取的信息，均包括一些指向其他网页的 URL，同样其他网页的信息中也包含指向本网页的 URL，因此不同的 URL 之间存在着一种循环指向的问题。如若对它置之不理，网络爬虫程序就会在这些 URL 之间循环抓取，比较严重的情况是两个 URL 相互指向对方，如果爬虫程序一直在这两个 URL 之间抓取信息，就会形成死循环。因此 URL 管理器有一个很重要的作用就是防止重复抓取和循环抓取网页。

（2）网页下载器

网页下载器的主要功能把网页对应的 URL 下载到本地，它是整个爬虫程序的核心组件。网页下载器和浏览器相似，它从互联网上下载 URL 对应的网页，将其内容按照 HTML 的格式下载，然后按照本地文件或者本地字符串的形式来存储，然后再进行后续的分析处理。

（3）网页解析器

将互联网上的 URL 下载到本地后，需要通过网页解析器对该 URL 进行解析才能够提取出所需要的内容。简而言之，网页解析器是从网页中提取人们需要的数据的工具。从一个搜索引擎来看，网页解析器首先会将网页中所有的 URL 提取出来，以便后续进行访问。本文所做的是一个定向爬虫，除了将网页中的待爬取的 URL 提取出来之外，还要将所需要和感兴趣的数据提取出来。即是说网页解析器会把网页下载器下载的 HTML 网页文档字符串作为输入来提取出需要的内容和未访问过的待爬取的 URL 列表。

（4）网页输出器

网页输出器实际是网页处理的一部分，抓取到网页的数据后，利用网页解析器提取出该网页中需要的数据，然后将这些数据写入本地的一个 HTML 文件中。如果想要对抓取到的数据进行其他的处理，就需要修改相应的代码，增加新的功能模块。在本课题研究中，主要是将爬取到的网页内容存储到本地的 HTML 文件中，网页输出器需要对外提供两个方法，其主要的方法是实现写入文件这个功能。

网络爬虫运行流程如图 2 所示，由爬虫的总调度程序来启动或停止爬虫，查看爬虫的运行情况。在爬虫程序中，URL 管理器用来管理待爬取的 URL 列表和已经爬取过的 URL 列表，从 URL 管理器中取出一个 URL，判断该 URL 是否被爬取过，如果是未被爬取的 URL，则将这个链接发送到网页下载器。下载器下载由 URL 链接指向的网页，并将下载下来的内容以字符串的形式存储下来，然后会把这个字符串提交到网页解析器，由网页解析器进行解析，会解析出我们所需要的数据。同时每个页面中都有指向其他页面的链接，通过网页解析器把它们都解析出来后，增添到 URL 管理器中。这三个部分共同构成了一个循环，只要有满足条件的 URL，程序就一直持续运行。

2　并行爬虫程序的实现

在串行网络爬虫的基础上，可以实现多线程的网络爬虫程序，当爬虫开始执行后，程序向网页发送访问网页的请求，然后

程序等待网页作出响应。等待时间越长，效率也就越低。当程序采用多线程时，交互消息期间的平均等待时间有所降低，可以提高数据抓取的效率。同样给定一个入口 URL，从这个入口 URL 的网页页面内容之中解析出所有的 URL 链接。如果这些链接没有被访问过，增添到待爬取 URL 的队列中，然后再从待爬取的 URL 列表中取出一条进行访问和解析。程序中需要增加一段创建线程池的代码，一开始给定一个最大线程数，每在待爬取的 URL 列表中取出一个 URL 时就添加一项任务入队列，执行任务时，就从队列中出取出一项任务并执行。

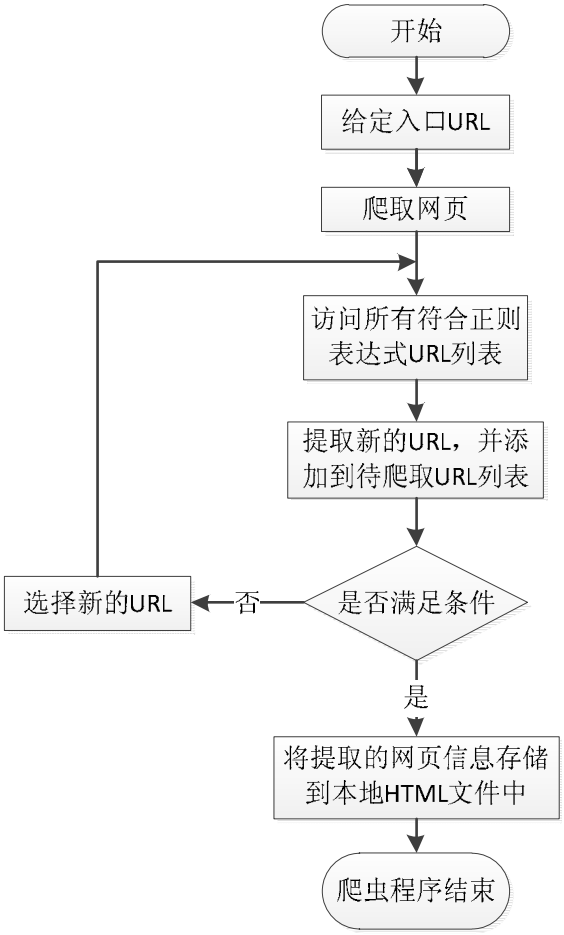

图 2 网络爬虫运行流程图

3 实验设计与分析

将网络爬虫程序的入口地设定为 http://stackoverflow.com/questions 网页，设定程序爬取 1000 个网页就停止运行，打印程序运行的时间，将串行爬虫程序和多线程爬虫程序的运行时间进行对比。以爬取网页的数量 100、300、500、800、1000 作为横坐标，以爬虫程序的运行时间（单位为 s）作为纵坐标，分别画出不同的网页规模下，线程个数分别为 1、3、5、8、10、15 的时候运行时间的变化。折线图如图 3 所示。

从图 3 中可以得出下面几点结论：

当爬取的网页数量为 100 的时候，串行爬虫程序的运行时间和在不同线程个数下爬虫的运行时间相差不大。当爬取的网页数量为 300 和 500 时，线程数量的增加也并没有让运行时间有显著性的提高。因此使用多线程的爬虫程序一般也只是在网站规模大，爬取的网页数量很多时才会有显著的提升效率的作用。

选定某一个网页规模观察数据，会发现线程数增加的时候运行的时间没有降低反而增加，是因为线程之间进行切换也需要耗费时间。因此增加线程数量并不是绝对的提高线程，根据程序运行的环境的不同，能够提高爬虫程序运行效率的最大线程的个数也不相同。

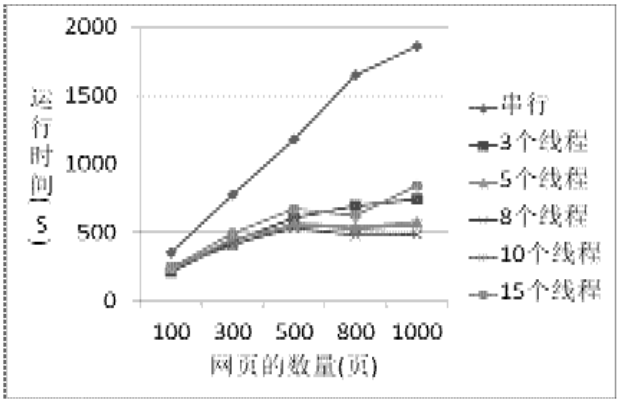

图 3 特定网页规模下线程数—爬虫运行时间折线图

4 总结与展望

通过本文的实验可以得出，在特定的条件下，通过增加爬虫程序的线程数能够提高网络爬虫的效率，但是在设置网络爬虫线程的时候也要考虑多种因素，比如说过多线程之间的切换所耗费的系统资源以及程序运行时所在网络情况[4]，而且有的网站会限制下载的速度，线程的数量太多时，大量线程访问网页，某些线程会被挂起。后期可以考虑在多台服务器上分布式[5] 部署网络爬虫，实现分布式爬虫之间的通信模式，进而提高网络爬虫的效率。

参考文献：

[1]薛建春.垂直搜索引擎中网络蜘蛛的设计与实现[D].北京:中国地质大学检测技术与自动化装置自动检测及应用, 2007.

[2]姜彬彪, 黄凯林, 卢昱江等.基于 Python 的专业网络爬虫的设计与实现[J].企业科技与发展(企业科技创新版), 2016.

[3][澳] Richard Lawson 著，李斌译.用 Python 编写网络爬虫[M].人民邮电出版社, 2016.

[4]阳国贵, 姜波.线程切换开销分析工具的设计与实现[J].计算机应用, 2010.

[5]王毅桐.分布式网络爬虫技术研究与实现[D].成都:电子科技大学信息安全, 2012.