文章编号:1001 - 9081(2008)05 - 1114 - 03

# 网络爬虫效率瓶颈的分析与解决方案

尹 江,尹治本,黄 洪
(西南交通大学 信息科学与技术学院,成都 610031)
(j_yeen@163.com)

**摘 要**:网络爬虫的效率,直接关系到搜索引擎系统为用户提的供服务质量。如何设计高效、快速的网络爬虫,成为目前网络爬虫研究的热点。要提高网络爬虫的爬行效率,除了需要改进网络爬虫的爬行策略之外,还需要优化网络爬自身的设计,改进网络爬虫自身的结构,消除效率瓶颈。通过对网络爬虫结构、应用环境以及用户要求的分析,提出一个通用网络爬虫的改进设计方案,并通过实验得到较好的测试结果。

**关键词**:爬行策略;套接字;多线程;网络爬虫
**中图分类号**: TP311 **文献标志码**:A

# Efficiency bottlenecks analysis and solution of Web crawler

YIN Jiang, YIN Zhi-ben, HUANG Hong
(*School of Information Science and Technology, Southwest Jiaotong University, Chengdu Sichuan* 610031, *China*)

**Abstract:** The efficiency of a web crawler determines the quality of services a web searching system offers to its users. How to design a more efficient and faster web crawler is becoming a hot issue in the research of web crawler. In order to raise the crawling efficiency of a web crawler, the crawling strategy needs to be reformed. Besides, the design of the web crawler system has to be optimized and its structure also needs to be improved to eliminate bottlenecks. In this paper, an improved scheme of designing a general web crawler was presented through analyzing crawler's structure, application environment and user requirement, and the preferable testing result has proven better efficiency it has.

**Key words:** crawl strategy; socket; multi-thread; Web crawler

网络爬虫是搜索引擎的重要组成部分。目前爬虫系统的基本设计原则为:在遵循 REP 原则以及对服务器不造成致命冲击的前提下[1],尽可能使爬虫爬行速度快、数据下载量大及信息抓取准确。必须要消除制约爬虫自身爬行效率的瓶颈,使爬虫达到高效。

## 1 网络爬虫简介

通用网络爬虫爬行的基本策略是将 Internet 视为一幅复杂的有向图。利用这样的模型,网络爬虫可以采用图的广度优先搜索算法或图的深度优先搜索算法爬行 Internet 并下载数据。

### 1.1 广度优先、深度优先爬行策略

一个网页即为一个节点,网页中指向其他页面的 URL 为该节点到其他节点的路径,整个 Internet 由大量这样的节点构成一幅庞大的有向图 $G(E,V)$,如图 1 所示。

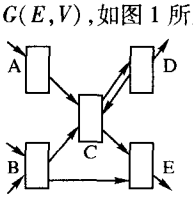



图 1 Internet 的有向图模型示意图

其中矩形代表页面,箭头线为 URL,该图显示了网页间相互链接的关系。无论是广度优先还是深度优先策略,其时间渐近复杂度都为 $O(e+v)$,其中 $v$,$e$ 分别为图的节点与边的数量,即与 Internet 中的网页规模直接相关。上述爬行策略对各个网站、页面和 URL 的价值回报并不评估筛选,爬行速度快但针对性较差,不能提高搜索的查准率。

### 1.2 基于价值回报的爬行策略

网络爬虫理想的设计是高速、完整地遍历整个 Internet。往往需要对单纯的图算法爬行策略进行改进,合理地对资源(网站、页面及 URL)进行价值评价,优先处理值高的资源,滞后处理甚至忽略价值低的资源。目前实际应用的策略主要有:基于链接自身质量评价的 PageRank 算法以及 HITS 算法、基于 URL 主题相关性评价的 Best Search 算法及 Fish 算法等[2]。除此以外机器学习理论、人工神经网络算法、蚂蚁算法等方法也在不断地应用到网络爬虫寻路优化策略中[3]。

## 2 爬虫的瓶颈分析与解决方案

### 2.1 效率瓶颈分析

爬虫的效率主要受到以下因素的制约:网络延时和爬虫本地运行效率,如图 2 所示。

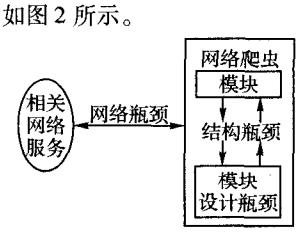



图 2 网络爬虫的效率瓶颈示意

网络爬虫最主要的效率瓶颈在于网络带宽利用率低、适应性差;功能模块设计不良;各个功能模块协同工作效率低下等。

目前绝大多数爬虫系统都采用并发工作流的设计,以充分利用网络带宽。由于基于进程的并发代价较基于线程的并

**收稿日期**:2007 - 11 - 12;**修回日期**:2008 - 01 - 04。

**作者简介**:尹江(1981 - ),男,四川成都人,硕士研究生,主要研究方向:计算机算法理论、软件工程; 尹治本(1954 - ),男,云南腾冲人,教授,主要研究方向:计算机算法设计、软件工程; 黄洪(1959 - ),男,四川达州人,副教授,主要研究方向:数据库、办公自动化。

发而言相对较高,故大部分网络爬虫都是多线程架构设计[4]。然而这并不能完全疏通爬虫的效率瓶颈。

### 2.2 网络资源利用率的提升策略

基于Socket(以下统称套接字)的网络爬虫使用套接字,通过发送HEAD、GET、POST等HTTP方法,爬虫能在HTTP协议上通过指定的端口与服务器进行数据信息交换[5]。爬行过程中爬虫需要两次使用网络资源:域名解析与页面采集,致使网络延时占据绝大部分爬虫运行时间,形成爬虫运行效率的瓶颈。在实际测试中,对100个主机名通过查询DNS服务器得到IP地址,平局时间为327毫秒/个。其中有少数域名的查询返回时间甚至超过数秒。同时,某些数据量大的网页的传输等待时间也会超过数秒。

#### 2.2.1 DNS解析

引入并优化DNS缓存模块。URL中重复的域名使用频繁,DNS本地缓存能大量减少因重复的域名解析造成的网络占用及等待时间。为提高域名缓存模块的效率,本文设计了一个使用哈希表为表头、以线性指针序列作为索引并以域名长度为跳跃单位的数据结构保存域名,暂命名为"域名跳检哈希表",能够高效的写入域名、检索域名、为域名排序以及高效地按需求替换域名。其表结构的一个环节如图3所示。

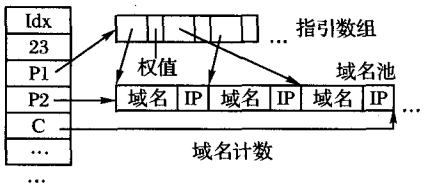



图3　用于DNS缓存的域名表结构

图3展示了域名表的构造与关键环节。改进后域名解析过程大致如下:使用域名首字符ASCII码值与域名长度散列域名到哈希表头。依照线性指针序列的下标索引,通过域名头指针依次检索已存在IP映射的域名,若该域名还未在表中则调用DNS解析过程。解析成功便将域名写入域名表最后空位,IP则写入对应IP段内,并更新域名池信息(包括权信息、数量信息等);失败则返回错误代码通知调用者。在写入时若发现域名池满则替换掉部分权值低的域名。若该域名已经过解析则使用对应IP,并对域名进行相应的加权(如使用频率、最近使用时间等)。为保证权值高的域名能够被快速地映射出IP,在若干次域名解析与写入过程后需要为域名排序。排序时以线性指针链索引遍历所有存在域名的权值,需要改变域名顺序时仅仅交换域名指针域与权值域。该结构兼有哈希表、链表与线性表的优点,下面是主要操作的算法时间效率分析:

插入域名:新域名$h$到达时,计算其HASH索引的时间为固定常数,计为$T_1$。由于域名池空位地址 = 域名池基址 + 域名个数 × 域名长度,故寻址域名池空位时间为固定常数$T_2$。另计域名的写入操作时间为$T_3 = t(l)$,$l$为域名长度。则可知一个新域名的插入时间复杂度为$T_1 + T_2 + T_3 \approx O(c)$。

域名排序:为域名按权值排序时仅仅做指针交换操作,大大优于单纯的线性表结构。设某个域名池存放长度为$n$的域名$m$个,若单纯使用线性表结构操作则每次移动一个域名需要移动$n$个元素3次,若每个元素都需换位且仅需1次,则至少需要$3nm$次移动操作,而在本文所采用策略下$m \approx 1$,即效率为普通线性结构的约$m$倍。

域名映射:新域名$h$到达时,根据其首字符编码以及$h$的长度$l$计算HASH索引,探测$h$可能存在映射的域名池的时间计为固定常数$T_1$。现在分析$h$在池中寻找匹配的平均时间$T_2$。设域名池已有$n$个域名,每个域名固定长度为$l$,$h$中第$i$个字符失配而前$i-1$个字符匹配的概率为$P_i$,$i = 1,2\cdots l$,又设$h$被某个域名完全匹配的概率为$P$,则有$P + \sum_{i=1}^{l} P_i$,且第$i$个字符匹配后已经比较过的字符数为$L_i$。设$P'_i$为$h$与域名池中前$i-1$个域名失配但与第$i$个域名匹配的概率。现做$N$次域名映射操作,则可知:

$$T_2 = \frac{N \times P'_1 \times \sum_{i=1}^{l} EX_i(L)}{N} + \frac{N \times P'_2 \times \sum_{i=1}^{2} EX_i(L)}{N} + \cdots + \frac{N \times P'_n \times \sum_{i=1}^{n} EX_i(L)}{N} \tag{1}$$

其中$EX_i = P \times L + \sum_{j=1}^{l} ((1 - P_j) \times L_j)$,$i = 1, 2, \cdots, n$为域名池中每个域名与$h$失配所移动的字符数的数学期望。该结构的优势体现在当池中域名某个字符与$h$中字符失配时,可以直接跳到下一个域名起始处比对,即每次映射操作比较字符数远小于$n \times l$,同时还可以加入模式匹配优化策略,域名越长,效果越好。

#### 2.2.2 页面采集

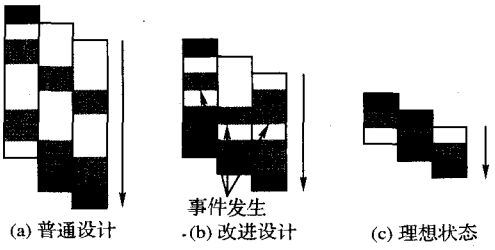
多线程、非阻塞套接字与WSAEventSelect(异步)模型的组合设计。核心思想是采用适应性更强的方法,最大限度利用网络资源[6],同时缩短线程执行周期。在采集页面的过程中,爬虫需要长时间等待数据到达协议缓冲区。若采用多线程并发爬行的设计,应开启多个爬行线程并让等待中的线程阻塞,既能充分地利用闲置的网络资源,又尽可能地减少了同时占有CPU的线程数量,缩短线程执行周期。虽然事件选择模型本身支持套接字组管理方式,但套接字组中的最大套节字数极为有限(64个),且必须维护线程池使系统达到高效。此外,套接字组管理增加了套接字行为的管理难度。本文采用每个异步套接字绑定一个工作线程的创新设计,线程队列在爬虫开始爬行前创建,在爬行过程中不会被撤销,无需线程池且读写操作不分离,既提高了效率又方便管理。具体实施方案如下:1)将套接字设定为非阻塞方式,并绑定在一个WSAEVENT对象上,通过探察这个对象的状态以获知发生了哪些需要处理的网络事件,如可读取、可发送、关闭连接等等。2)在没有相关的事件发生且不满足采集工作结束条件时,线程被阻塞一个超时。3)若在系统阻塞线程等待数据的过程中有数据到达,系统会唤醒线程继续读取所有到达数据,同时超时计数器复位。4)否则超时计数器加1,继续探察事件对象。同时每次阻塞前首先检查采集工作结束条件(如超时计数器为0、对方关闭连接等以及文件已结尾),判断是否中止数据读取操作,尽可能缩短线程执行周期。通过此种设计,一方面线程因等待数据阻塞时,CPU得以尽可能多地执行有效运算;同时,通过事件机制,使得套接字工作能适应更加复杂的网络环境。

图4为爬行线程工作队列的时间片分布示意图,图中每组矩形表示一个爬行线程工作队列,其竖直方向的长度显示了一个页面采集过程的周期长度。矩形中的灰色部分为线程阻塞时间,白色部分为多个线程共享的CPU时间,黑色部分为线程独占的CPU时间,线程队列旁的箭头线长短表示线程

的执行时间。图4(c)显示了一种理想状态(规定线程必有一次阻塞):每个线程的CPU时间独享,且阻塞的时间最短并只阻塞一次。从图4(b)中可以看出,由于事件机制能及时唤醒阻塞中的线程,减少了线程的不必要的阻塞时间。设 $T_{1i}$ 为某页面分次传输的真实耗时,并且发生 $m$ 次。又设 $T_{2i}$ 为人工设定超时上限,超时等待次数为 $n$ 次。基于下面的事实:(1) 超时等待总时间必须大于或等于页面传输真实耗时才可能正确的下载页面;(2) 每次数据到达前人工的超时等待必至少发生一次;(3) 探查到数据未到达后的等待超时应至少等于页面传输时间[7]。则有对任意的 $i, n \geqslant m, T_{2i} \geqslant T_{1i}$,可知浪费的等待时间为:

$$T = \sum_{i=1}^{m} (T_{2i} - T_{1i}) + \gamma \times \sum_{i=m+1}^{n} T_{2i} \quad (2)$$

其中引入文档结束标志检测机制时 $\gamma$ 概率等于0,否则等于1。通过优化设计,由于事件通知机制会使得 $T_{2i}$ 逼近 $T_{1i}$,使得方程右边第一项远小于普通设计方式下的结果,大大缩短单次页面采集周期。




(a) 普通设计 (b) 改进设计 (c) 理想状态

图4 改进机制的效率提升示意

### 2.3 爬虫本地运行效率的优化方案

除网络资源外,爬虫自身各部分的运行效率也可能成为爬虫工作效率的瓶颈。

多线程工作同步是爬虫系统正常工作的必要前提[8],但大量工作线程同步意味着排队等待时间增加,在共享数据操作频繁的环境下,系统工作效率甚至会因线程数量的增加而下降,同时还会带来大量的系统开销来实现临界区操作,造成效率瓶颈。本文采用URL队列独享,URL散列结构共享的结构设计。实际测试发现,URL队列是整个爬虫中访问最频繁的部分,应尽量避免同步问题。现有线程工作队列 $P_1 \cdots P_n$,若其中有一半的线程在做 $m$(所有线程的平均值)个URL入队列操作,并且其中有20%的操作重叠,另设平均一次入队列操作时间为 $t$,假定CPU线程调度均匀(此时线程入队列操作排队等待时间平均分摊到每个线程上),则得到同步等待时间,如式(3):

$$T_i = \alpha \sum_{k=1, k \neq i}^{n} (\beta m_k t_k) \quad (3)$$

其中 $\alpha$ 为试图访问临界区线程的比例,$\beta$ 为入队列操作的平均重叠率,$m_k$、$t_k$ 分别由平均值 $m$、$t$ 取代。按上述条件粗略地计算出线程 $P_i$ 在URL入队列的过程中,由于同步浪费的等待时间为 $T_i = nmt/10$。由此可看出每个线程包含自己的URL队列是非常合理的。另一方面,URL散列结构必须共享,原因是URL消重效果不能牺牲,若作为线程独立的结构,需要大量额外的时间、空间上的开销来为每个线程同步URL消重散列结构的数据。其次,URL消重操作较为分散(本文设计的爬虫消重工作只在页面采集过程前端进行),操作时间短且各线程的重叠操作很少,对整个工作队列的运行效率影响不明显。

## 3 测试与小结

综合以上论述,笔者在Visual Studio 6.0环境下用C++语言开发了一个工作在Windows系统上采用广度优先策略的通用爬虫,主要目的在于测试在选定爬行策略的前提下,爬虫自身设计的改进以及主要瓶颈的消除所带来的爬行效率提升。测试环境如下:Intel P4 2.8 GHz(CPU);DDR400 1 GB(内存);7200 Rpm 80 GB 串口(硬盘);Windows XP(操作系统);校园网网通(网络)。该系统结构如图5所示。

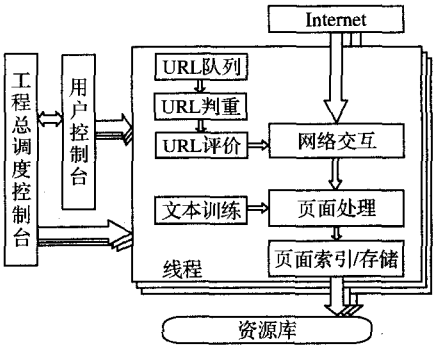



图5 SPIDER爬虫系统结构

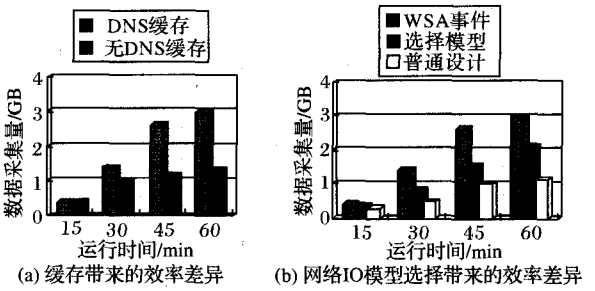
通过该系统对DNS缓存模块的引入、网络交互模型选择、并发优化阈值以及URL队列构造策略等对爬虫效率的影响进行测试。

表1 三大门户网站首页下载测试数据(2007年6月7日)

| 页面 | 大小/kb | 连接次数 | 保持连接 | 采集周期/s 普通设计 | 非阻塞探查 | 组合优化设计 |
|---|---|---|---|---|---|---|
| 新浪首页 | 308.5 | 50 | 是 | 3.20 | 2.85 | 2.25 |
| 网易首页 | 267.5 | 50 | 是 | 2.97 | 2.90 | 1.07 |
| 搜狐首页 | 234.6 | 50 | 否 | 3.02 | 2.62 | 0.93 |

表1为对三大门户网站的首页的下载采用不同设计所得到的结果比较。可以看出,在Server不与Client保持长连接时,优化效果最为明显,采集周期缩短近70%;而保持长连接的情况中,若引入文档结束检查机制,也有颇为明显的改善。

图6显示了DNS缓存的引入及WSA事件机制对爬虫效率的影响,其中横坐标表示爬虫的运行时间,以15 min为单位间隔;纵坐标为爬虫的数据采集量,以千兆字节计。可以看到,引入DNS缓存使爬虫效率提升了近两倍,而事件选择模型与套接字绑定工作线程的组合设计也大大提升爬虫的爬行效率,达到了设计目的。




(a) 缓存带来的效率差异 (b) 网络IO模型选择带来的效率差异

图6 测试数据比较

表2列出了不同结构的爬虫在本文所述测试环境下爬行30 min所测得的关键综合数据。从上面的数据中,还可以看到URL队列共享对爬虫工作效率的负面影响也颇为明

(下转第1119页)

而不是如 LEACH 那样随机地轮循簇首。充分说明 EDBCM 协议提高了网络的能量有效性,能提供更多数据来刻画传感区域,更好地完成网络所担负的任务。

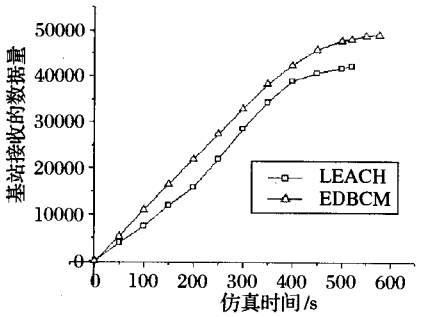



图4　LEACH 和 EDBCM 基站数据量比较

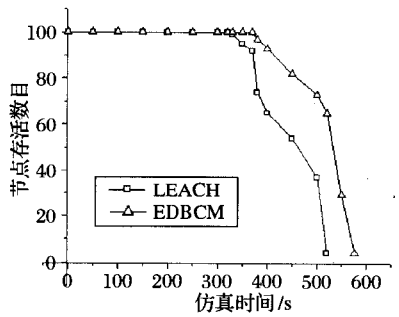



图5　LEACH 和 EDBCM 节点存活数比较

从图5可以看出,EDBCM 中第一个节点死亡的时刻为370 s,LEACH 为320 s,比 LEACH 延后了15.6%;第20个节点的死亡时刻为460 s,LEACH 为375 s,延后了22.7%。与 LEACH 相比,EDBCM 中节点死亡的时刻明显延后。让剩余能量较多、距离基站较近的节点担当簇首,有效地保护了能量较低的节点,使节点间的剩余能量差别不大。另外,多跳的数据转发路由,减少了通信能耗,同样也延缓了节点的死亡时间,有效提高了传感器网络的工作寿命。

## 4　结语

本文提出一种基于能量和距离分簇的多跳路由协议。簇首选择时充分考虑了节点能量和到基站的距离,簇首质量较高;数据发送采用了改进的多跳路由。仿真结果表明,与 LEACH 协议相比,该算法提高了基站接收的数据量,明显延长了网络的生存寿命。今后,可用 MATLAB/OPNET 在大型网络中做进一步的仿真测试。另外,数据转发过程中潜在的数据包丢失和时延问题,也是要研究的问题。

**参考文献:**

[1] 宋文,王兵,周应宾,等. 无线传感器网络技术与应用[M]. 北京:电子工业出版社,2007.

[2] HEINZELMAN WR, CHANDRAKASAN A, BALAKRISHNAN H. An application-specific protocol architecture for wireless microsensor networks[J]. IEEE Transactions on Wireless Communications, 2002, 1(4): 660 - 670.

[3] LINDSEY S, RAGHAVENDRA C, SIVALINGAM K M. Data gathering algorithms in sensor networks using energy metrics[J]. IEEE Transactions on Parallel and Distributed Systems, 2002, 13(9): 924 - 935.

[4] MANJESHWAR A, AGARWAL D P. APTEEN: A hybrid protocol for efficient routing and comprehensive information retrieval in wireless sensor networks[C]// Proceedings of the 16th International Parallel and Distributed Processing Symposium(IPDPS 2002). Washington, DC: IEEE Computer Society, 2002: 195 - 202.

[5] ZHANG HAI-BO, CHEN DI, Lowest energy protective clustering algorithm for wireless sensor networks[C]// International Conference on Sensing, Computing and Automation (ICSCA2006). Chongqing: [S. n.], 2006: 2856 - 2859.

[6] HEINZELMAN W, CHANDRAKANSAN A, BALAKRISHNAN H. Energy-efficient communication protocol for wireless mircrosensor networks[C]// Proceedings of the 33rd Hawaii International Conference on System Sciences (HICSS'00). Washington, DC: IEEE Computer Society, 2000: 3005 - 3014.

[7] ZHANG WEN-YA, LIANG ZI-ZE. A power efficient routing protocol for wireless sensor network[C]// Proceedings of the 2007 IEEE International Conference on Networking, Sensing and Control (ICNSC'07). Washington, DC: IEEE Computer Society, 2007: 20 - 25.

(上接第1116页)

显。另外,为了进一步提高爬虫在本地的运行效率,还需要找出并发工作线程数量在某个确定运行环境下最优的阈值。

表2　爬虫的测试数据比较

| 阻塞模式 | I/O模型 | DNS缓冲 | 线程数量 | 队列独享 | 下载速度 page/s | 数据下载量/GB |
|---|---|---|---|---|---|---|
| 否 | 事件 | 是 | 30 | 是 | 26 | 1.41 |
| 否 | 事件 | 是 | 30 | 否 | 5 | 0.33 |
| 否 | 事件 | 是 | 20 | 是 | 13 | 0.69 |
| 否 | 事件 | 否 | 30 | 是 | 16 | 0.95 |
| 否 | 选择 | 是 | 30 | 是 | 14 | 0.79 |
| 是 | 无 | 否 | 30 | 是 | 6 | 0.47 |
| 是 | 无 | 否 | 30 | 否 | 3 | 0.21 |

## 4　结语

网络爬虫的策略选择不当以及自身结构设计不良,都会给爬虫工作效率造成不良影响。通过改进模块本身设计及协调各个模块的工作等方法,可以消除部分爬虫系统工作效率的瓶颈,提高爬虫系统的爬行效率。目前对网络爬虫系统的研究正在不断深入,许多针对爬虫爬行效率提升的改进方案也不断被提出并被广泛采用。

**参考文献:**

[1] 苗长芬,冯伟华. 面向主题 Crawler 的设计与实现[J]. 平原大学学报,2005,22(3):110 - 112.

[2] 黄河燕. 基于增量反馈和自适应机制的主题爬虫系统的设计与实现[D]. 南京:南京理工大学,2005.

[3] 刘金红,陆余良. 主题网络爬虫研究综述[J]. 计算机应用研究,2007,24(10):26 - 29.

[4] 陈杰. 主题搜索引擎中网络蜘蛛搜索策略研究[D]. 杭州:浙江大学,2006.

[5] BEHROUZ A F. TCP/IP Protocol Suite[M]. 2nd ed. 谢希仁,译. 北京:清华大学出版社,2003.

[6] 李晓明,闫宏飞,王继明. 搜索引擎—原理、技术与系统[M]. 北京:科学出版社,2004.

[7] 朱玉丽. 基于网格技术的主题爬虫算法优化的研究与实现[D]. 沈阳:沈阳工业大学,2007.

[8] 何世林. 基于 Java 技术的搜索引擎研究与实现[D]. 成都:西南交通大学,2006.