分布式网络爬虫的设计与实现

吴黎兵[1,2] 柯亚林[1,2*] 何炎祥[1,2] 刘 楠[2]
[1](武汉大学计算机学院 湖北 武汉 430072)
[2](信息网络安全公安部重点实验室 上海 201204)

摘 要 提出一种可部署于单一网域及多网域间的分布式爬虫 DSpider。DSpider 能够通过调整节点规模和连接超时阈值,有效部署于 LAN 和 WAN 两种网络环境中。首先简要介绍了 DSpider 的系统结构,然后详细分析了 DSpider 的任务调度策略,并且在实验中将 DSpider 爬虫部署在 LAN 和 WAN 两种环境中的不同性能作了详细的分析。

关键词 DSpider 系统架构 节点协同 Web 划分

中图分类号 TP393 **文献标识码** A

DESIGN AND IMPLEMENTATION OF A NOVEL DISTRIBUTED WEB CRAWLER

Wu Libing[1,2] Ke Yalin[1,2*] He Yanxiang[1,2] Liu Nan[2]
[1](*School of Computer, Wuhan University, Wuhan 430072, Hubei, China*)
[2](*Key Lab of Information Network Security, Ministry of Public Security, Shanghai 201204, China*)

Abstract In this article, a novel distributed web crawler DSpider is presented. DSpider can be deployed in single network domain and among multiple network domains, by adjusting its number of nodes and the threshold of connection timeout, it can also be effectively deployed in two network environments of both LAN and WAN. In the article, firstly the system architecture of DSpider is introduced briefly. Then the task scheduling strategy of DSpider is elaborately analysed. The article also gives a report of the experiment in which different performances of the DSpider disposing in two environments of LAN and WAN are analysed in detail.

Keywords DSpider System architecture Node collaboration Web partition

0 引 言

作为搜索引擎和网络数据挖掘的基础组成部分,网络爬虫(Web Crawler)起着重要的作用。随着网络规模和复杂程度的爆炸性增长,依赖计算机单机处理能力的集中式网络爬虫无法满足快速获取大量资源的需求。分布式网络爬虫由可并行获取资源的多个 Agent 组成,具有良好的可扩展性,能满足人们对资源检索的实际需要。这些 Agent 部署的地理位置和网络拓扑可以不尽相同,因此可将集中部署在同一个局域网 LAN(Local Area Network)内部的称为单一域网络爬虫(Single-domain Distributed Crawler,或称局域网爬虫),将分散部署在广域网 WAN(Wide Area Network)不同位置的称为多网域网络爬虫(Multi-domain Distributed Crawler,或称广域网爬虫)。单一域网络爬虫的规模和速度受局域网网络总带宽的影响。多网域网络爬虫规模受网络带宽影响较小,但节点间通讯时延不可靠,检索效果易受地理位置和网络结构的影响。

1 技术现状

近几年来,分布式网络爬虫是对 Web 数据挖掘工具研究的主流。在学术方面,Cho 等人[6]首次给出了分布式爬虫的分类方法、评价指标等一系列基本概念,并提出基于广域网分布式爬虫与部署于 LAN 的系统相比,具有高可扩展性和减少 Internet 负载的优点,为广域网分布式爬虫的研究奠定了基础。UbiCrawert[7]扩展了文献[6]中的一些概念,并声称可以支持基于广域网的分布式平台。

国内学术界对分布式爬虫研究也越来越多,代表性的有北京大学天网搜索引擎的爬虫系统[8],这是一个基于 LAN 的爬虫,已经开始商业化运作;上海交通大学的 Igloo 爬虫实现了基于网格服务的分布式爬虫(IglooG)[9]。但大多数研究都为基于局域网环境的单一域网络爬虫,多网域网络爬虫的研究成果较少。本文提出一种可部署于单一网域及多网域间的分布式爬虫 DSpider,简要介绍了 DSpider 的系统结构,详细分析了 DSpider 的任务调度策略,通过实验证明 DSpider 爬虫可通过调整节点规模和连接超时阈值,有效部署于这两种网络环境。

2 DSpider 系统架构

根据整个系统中各节点(Agent)间通信方式的不同,可将分布式网络爬虫划分为两种系统架构:主从式(图 1(a))和并列式

收稿日期:2011-09-01。2011 中国计算机大会论文。信息网络安全公安部重点实验室开放课题(201007001);湖北省自然科学基金(ZRY1496);中央高校基本科研业务费专项资金(6081013)。吴黎兵,教授,主研领域:计算机网络和分布式计算。

(图 1(b))。

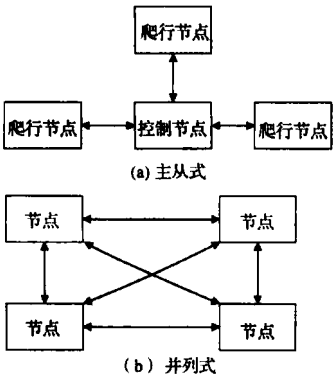

图 1　系统架构

主从式是指将整个系统中的 Agent 分为控制节点和爬行节点两种类型。控制节点负责爬行任务调度的工作,爬行节点负责抓取 Web 信息的任务。只有控制节点和爬行节点间发生通信,爬行节点间无通信。并列式是指整个系统当中的每个 Agent 地位相同,既负责抓取 Web 信息的任务,又要负责协调整个系统的任务调度的工作。每个 Agent 都维护一个地址列表,列表中存储着其他 Agent 的位置,可将爬行任务分配给其他合适的 Agent。当系统中的 Agent 数量发生变化时,每个 Agent 的地址列表都需要进行更新。二者的优缺点如下:

(1) 主从式架构的分布式网络爬虫任务调度系层次清晰,而且效率很高。但是主从式架构系统过分依赖于控制节点,一旦控制节点发生故障,整个系统将无法正常运行,当爬行节点数量远大于控制节点时,控制节点会成为整个系统的瓶颈。

(2) 并列式架构的分布式网络爬虫整体稳定性较好。系统中某个节点发生故障不会对系统性能产生大的影响。在任务调度中各节点之间要进行大量频繁的交互,占用了大量的网络带宽和计算资源。随着 Agent 数量的增加,用于检索资源的带宽降低,系统的运行效率较低。另外,由于节点间通讯时延,并列式架构同步地址列表耗时长于主从式架构。

为使 DSipder 分布式爬虫在 LAN 环境和 WAN 环境皆可部署,因此采用主从式架构,主要考虑以下两点:

(1) *资源利用率*　爬虫抓取 Web 信息时,网络带宽是极其重要的资源,它直接影响到爬虫系统的运行效率,并列式架构将大量的带宽消耗在节点的交互上,降低了资源利用率。

(2) *负载均衡*　主从式爬虫系统的任务调度集中在控制节点上,因此可以很方便地根据各个爬行节点的具体情况进行任务调度,有效控制各爬行节点的负载。而并列式架构难以对各节点的任务调动进行整体控制,各节点间任务分配不均,易导致部分爬行节点长期空置,浪费计算资源。

3　DSpider 任务调度策略

网络时延、带宽和机器性能是影响爬虫性能的三个制约要素。分布式爬虫系统的可扩展性使得它相对于单机网络爬虫在克服这三个要素上有着天然的优势。但分布式网络爬虫也有着自己的难题,比如如何避免节点之间抓取重复的网络文件,如何为爬行节点分配相对网络时延最小的任务。这一节将介绍 DSpider 的 Web 划分、节点协同和任务调度算法。

3.1　Web 划分

首先引入 DSpider 中 Web 信息单位的定义。

互联网上的文件采用 URL(网络资源统一定位符)标识。URL 由 4 个部分组成:网络协议、主机名、端口号、文件位置。DSipder 将具有相同主机名(host)的 URL 标识的文件集合作为最小的 Web 信息单位,其原因如下:

(1) 相同主机名的 Web 文件的网络传输时延基本相同,有利于将时延相同的 Web 文件让同一个爬行节点抓取。

(2) 如果一个爬行节点抓取一个主机上的文件因为连接超时失败,根据准则 1,它就可以放弃抓取该主机上的所有文件。这样有利于爬虫系统的任务管理。

(3) 同一主机上的文件平均语义耦合度相对较高。将同一主机上的文件存储于一个爬行节点上有利于爬虫对文件作进一步的处理。

Web 划分是指对 Web 资源进行划分,让不同的爬行节点抓取不同网络位置的 Web 文件。为了最大程度地优化下载性能,必须尽量让分布式爬虫系统中的每个爬行节点抓取相对于自身时延最小的 Web 文件。DSipder 引入了一个连接超时阈值 timeout,即爬虫系统中每个字节点在抓取一个 Web 文件时,一旦超过 timeout 时间还没有与服务器建立连接,就放弃抓取该 Web 文件。同时每个爬行节点有两张表:succeedhost 和 failedhost,控制节点有一张表 nodeshostmap。succeedhost 用来存放爬行节点抓取成功的主机名,failedhost 用来存储爬行节点抓取失败的主机名,nodeshostmap 用来存储所有爬行节点的抓取成功的主机名。图 2 是 DSpider 采用的 Web 划分策略,其中 controlunit 是指控制节点,node 是指爬行节点。

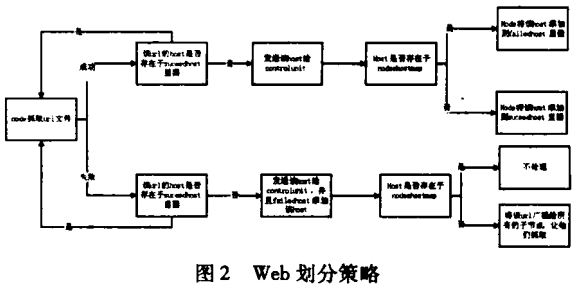

图 2　Web 划分策略

尽管 DSpider 通过 Web 划分策略将 Web 文件动态分配给时延相对较小的爬行节点抓取。但是还存在几个问题:首先,如何避免爬虫系统不同爬行节点抓取相同的 Web 文件;其次,一旦某个主机与所有爬行节点的时延都超过了 timeout 阈值,那么控制节点将不停广播该主机上的 URL。这两个问题可以通过节点协同解决。

3.2　节点协同

节点协同是指分布式爬虫系统的各子节点协同合作,共同完成 Web 资源下载任务。分布式网络爬虫在抓取 URL 的过程中会不断发现新的 URL,而这些 URL 存在大量的重复。如果将这些重复的 URL 直接交给发现它们的爬行节点去抓取,这势必引起多个爬行节点下载相同的网络资源,从而降低了爬虫系统的抓取效率。因而避免多个爬行节点抓取相同的 Web 文件是节点协同的首要问题。

3.1 节讲到每个爬行节点有两个表,succeedhost 和 failed host。为了控制爬行节点之间不抓取相同的网页,DSpider 中的每个爬行节点添加了一张表 visitedURLs,用来记录该爬行节点已抓取的 URL。图 3 展示 DSpider 的节点协同调度策略,其中 controlunit 是指控制节点,node 是指爬行节点。

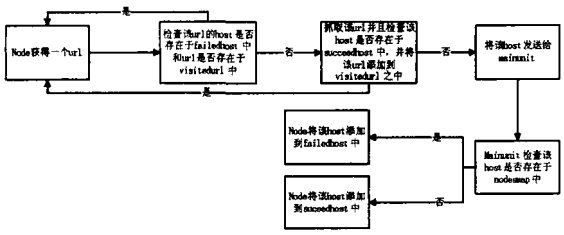

图3 节点协同策略

DSipder 通过节点协同策略解决了 3.1 节尾提到的关于节点协同的两个问题。通过 Web 划分和节点协同任务策略，DSipder 解决了分布式爬虫系统中的两个重要问题：如何避免多个爬行节点抓取相同的网络资源，如何让每个爬行节点抓取相对于自身而言网络时延最小的 Web 资源，以实现资源最优化配置。

3.3 任务调度

结合 Web 划分和节点协同，DSidper 进行任务调度的数据结构如下：

控制节点的 nodeshostmap 采用类似于 hash 链表的数据结构，如图 4 所示。

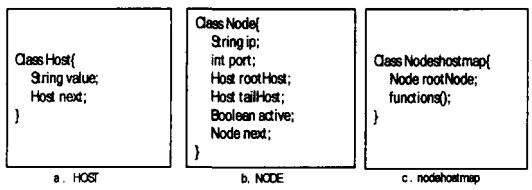

图4 nodeshostmap 数据结构

nodeshostmap：控制节点用来存放爬行节点信息的数据结构，rootNode 指向 nodeshostmap 中第一个爬行节点；functions( ) 提供了对爬行节点的插入、删除、查询等操作和对主机名的插入、删除、查询等操作的接口。

NODE：描述单个爬行节点信息的数据结构，ip 指爬行节点的 ip 地址；port 指爬行节点的爬虫程序占用端口；rootHost 指向该节点抓取的 Web 信息的主机名集合的第一个主机名；tailHost 指向指向该节点抓取的 Web 信息的主机名集合的最后一个主机名；active 标识该节点是否已经推出爬虫系统；next 指向 NodesTable 中该爬行节点的下一个爬行节点。

HOST：value 指主机名值；next 指向下一个 Host。

爬行节点的 visitedURLs 也采用类似 hash 链表的数据结构，如图 5 所示。succeedhost、failedhost 采用同步队列的数据结构。

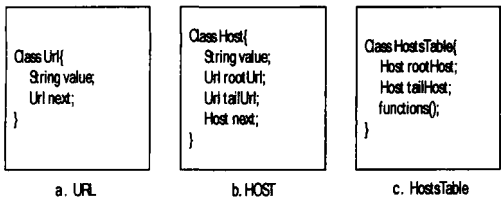

图5 visitedURLs 数据结构

URL：value 指 URL 的值；next 指向爬行节点抓取的与该 URL 相同主机名的 URL。

Host：value 指一个主机名的值，rootURL 指爬行节点抓取的主机名为 value 的 URL 集合的第一个 URL；tailURL 指爬行节点抓取的主机名为 value 的 URL 集合的最后一个 URL；next 指向爬行节点抓取的主机名集合中的该主机名的下一个主机名。

HostsTable：rootHost 指向该爬行节点抓取的主机名集合中的第一个主机名；tailHost 指向该爬行节点抓取的主机名集合中的最后一个主机名；functions( ) 提供了爬行节点查询、插入、删除 URL 和 Host 的接口。

爬行节点与控制节点之间的交互请求在 DSpider 中被称作 Host 请求。图 6(a) 指控制节点的 Host 请求，图 6(b) 指爬行节点的 Host 请求。其中 node 表示爬行节点，controlunit 表示控制节点。

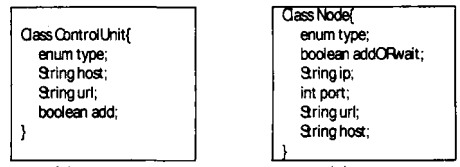

图6 Host 请求数据结构

图 6(a) 中 type 表示请求类型，add 为 true 表示控制节点希望爬行节点抓取该 Host 请求的 URL，add 为 false 表示爬行节点将 Host 请求的 host 值添加到 failedhost 中。图 6(b) 中，type 表示请求类型，addORwait 为 true 表示爬行节点希望控制节点节点将该 Host 请求的 host 值添加到 nodeshostmap 中，addORwait 为 false 表示爬行节点抓取该 Host 请求的 URL 失败，请求控制节点做下一步的处理(是否要发给其他节点等)。

图 7 展示了 DSpider 控制各个爬行节点协同工作抓取 Web 信息的算法伪码。其中 Node 表示爬行节点，controlunit 表示控制节点。

```
node

Capture a url:
1. get a url from todourls
2. if(the url already exists in visiteduris or the host of the url exists in the othervisiteduris)
3.   goto 1;
4. else capture the file of the url
5.     if (capture succeed) check whether the host of url exists in visiteduris;
6.       if(yes) add the url to visiteduris;
                 goto 1;
7        if(no) send the url and the host to the mainunit;
                 goto 1;
8     if(capture failed) check the reason
9        if(the reason is timeout) check the host of the url exists in the visiteduris
10           if(yes) goto 1;
11           if(no) send the url and the host to the mainunit;
                 goto 1;
12       else goto 1;

Receive a Host requese from mainunit:
1. if(true == add of the Host) add the url of the Host to todourls;
2. if(false == add of the Host) add the host of the Host to othervisitedhost;
```

(a) node 爬行节点任务调度算法

```
control unit

Receive the Host request from a node:
1 if(the host of the Host request exists in nodestable)
     send the host of the Host request  to the node;
2 else
3.    if(true == addORwait of the Host request)
        add the host of the Host request to nodestable;
4.    else
        send the url of the Host request to all nodes except the node;
```

(b) *controlunit* 控制节点任务调度算法

图7 任务调度算法

4 仿真实验

4.1 实验环境介绍

实验使用 8 台计算机部署 DSpider，1 台计算机设为控制节

点,7 台计算机设为爬行节点,处于一台百兆交换机下;1 台计算机模拟互联网出口,部署在另一交换机,该计算机模拟互联网上共 totalfile 个文件,平均分布于于 n 个主机($a_1, a_2, \cdots, a_n$)中,同时将文件的大小量化为 10 个等级(1f,2f,…,10f)。在 LAN 环境下 7 台计算机在一个网域,在 WAN 环境下将 7 台计算机划分至不同子网。在 LAN 环境下采用每个主机对各爬虫请求延迟相同,各主机延迟不同的仿真策略;在 WAN 环境下采用同一主机对不同爬虫请求延迟不同,各主机延迟不同的仿真策略。主机延迟量化为 9 个等级(1T,2T,…,9T),用 C_{ij} 表示不同爬行节点和不同主机之间的连接时延,如 $C_{ij}=37$ 表示 a_i 主机和 b_j 爬行节点之间的网络传输时延是 37。假定 C_{ij} 在(1T,2T,…,9T)区间内均匀分布。由于爬虫系统每抓取一个 URL,它就会解析该 URL 对应网页而得到新的 URL,这就使得爬虫系统以 change 概率外链到其他主机上的 URL。网络带宽则是指单位时间 T 内下载的最大文件单位数(f)。每台计算机的初始列表包含 m($m \in n$)个主机,他们每抓取一个文件,就有 change 概率获得($a_1, a_2, \cdots, a_n$)中的其他主机名,爬行直至完成所有主机的遍历。实验运行到每台计算机下载速度为 0 或抓取文件数为 totalfile 时停止。

由于爬虫系统部署在 LAN 和 WAN 两种环境时有着各自的特点。LAN 环境下的爬虫系统(LAN-DSpider)总带宽大小是固定的,WAN 环境下的爬虫系统(WAN-DSpider)爬行节点带宽固定,但是总带宽却是可扩展的。虽然某时刻节点间通信速率不可确定,但单位时间内网络总通信流量和爬行节点平均 I/O 通信流量是可知的,在单位时间内,我们用 LTB(Lan Total Bandwidth)描述 LAN-DSpider 总带宽,用 WAB(Wan Average Bandwidth)描述 WAN 下爬虫爬行节点的平均带宽。

实验从资源利用率、下载速度、控制节点与爬行节点交互频率、文件重复率四个角度分析 DSpider 在节点数、timeout 超时阈值发生变化的性能表现。

(1) 资源利用率(srcuse):分布式爬虫系统平均每一个节点单位时间 T 内所抓取的文件单位数 f。

(2) 下载速度(speed):整个分布式爬虫系统单位时间 T 内抓取的文件单位数 f。

(3) 控制节点与爬行节点交互频率(communicate):指单位时间 T 内控制节点收到和发出的 Host 请求个数。

(4) 重复文件概率(duplicate):爬虫系统下载的重复文件数占它下载的文件总数的比例。

4.2 实验 1

图 8 描述了爬行节点数量 n(3 - 7)发生变化,对 LAN-DSpider 与 WAN-DSpider 的性能影响。注意 speed 单位为 f/T,duplicate 单位为 1/10000(%%),srcuse 单位为 f/10T,communicate 单位为 1/100T。它们在实验 2 也使用相同的单位。

LAN-DSpider 环境:totalfile 200000,change 1%,n 400,m 3,timeout 5T,LTB 25f/T。

WAN-DSpider 环境:totalfile 200000,change 1%,n 400,m 3,timeout 5T,WAB 5f/T。

WAN-DSpider 文件重复率 duplicate 增长速度很快,且远大于 LAN-DSpider 下的 duplicate 值,这是因为 WAN-DSpider 下的控制节点与爬行节点的通信时延较大,这使得 WAN-DSpider 爬行节点发现新的主机名报告给主机需要更多的时间。如果该主机名已经被其他爬行节点抓取过,主机反馈这一结果时,爬行节点已经抓取了该 web 主机上的一些文件。同时 LAN-DSpider 与 WAN-DSpider 的 speed 值都随着爬行节点数量的上升而增加,但是 WAN-DSpider 的 speed 增长速度要略大于 LAN-DSpider,这是因为 LAN-DSpider 的总带宽是有上限的。同时 WAN-DSpider 下的 srcuse 随着爬行节点数量上升而略有增加,LAN-DSpider 则略有下降,这是因为 WAN-DSpider 的总带宽会随着爬行节点的数量上升而增加,而 LAN-DSpider 的总带宽会保持不变,单个爬行节点的带宽会随之下降。

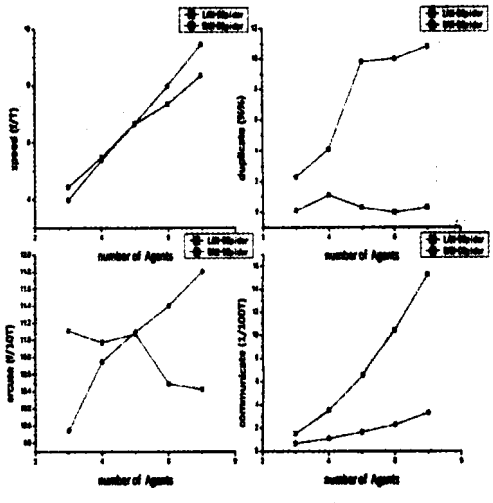

图 8　实验 1

4.3 实验 2

图 9 描述了连接超时阈值 timeout 发生变化(3T - 7T),对 WAN-DSpider 和 LAN-DSpider 性能的影响。

LAN-DSpider 实验环境:totalfile 200000,change 1%,n 400,m 3,LTB 25f/T,number of Agents 3。

WAN-DSpider 实验环境:totalfile 200000,change 1%,n 400,m 3,WAB 5f/T,number of Agents 3。

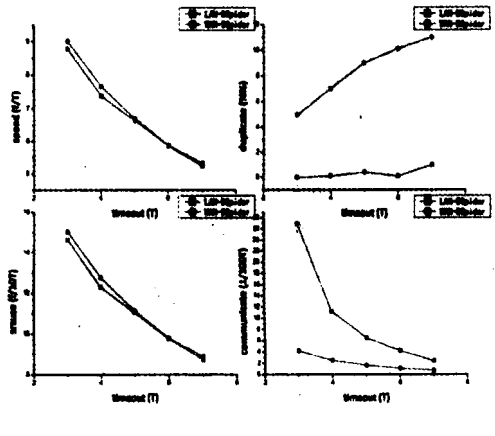

图 9　实验 2

同实验 1,在实验 2 中,WAN-DSpider 的 duplicate 值普遍大于 LAN-DSpider。随着 timeout 阈值的增加,WAN-DSpider 和 LAN-DSpider 二者的 srcuse、speed 都略有下降。这是因为爬虫用在与超时文件建立连接上的时间增加了。LAN-DSpider 与 WAN-DSpider 的 communicate 值都有所下降,但是 LAN-DSpide 下降幅度更大。

4.4 实验结论

通过在 LAN 和 WAN 环境下对 DSpider 爬虫的仿真实验,可确认该爬虫在两种网络环境下都能够有效运行。由于爬行节点

(下转第 213 页)

5 结 语

通过分析群智能算法框架的特点,引入基于本体的语义关系算子,进而提出基于语义关系的群智能算法,从算法上分析其在粒子群算法中的应用,最后在 TSP 问题上进行测试。实验结果表明:针对小规模数据集,基于语义关系算子的粒子群算法的寻优能力比经典算法有明显提高,同时减少了迭代次数。基于语义关系算子的群智能算法不仅可以在较短时间内找到最优值,在寻优的过程中扩大了搜索范围,避免陷入局部最优点,增强了算法的可靠性。由于基于语义关系算子的群智能算法还需更多的验证工作,算法的理论基础的研究和相关的应用都可以促进算法的成熟.算法的效率及收敛性也是值得进一步研究的课题. 综上所述,基于本体思想的语义关系群智能算法在求解复杂优化问题时优势明显,可以有效地避免群体陷入局部最优,同时也具有较快的收敛速度,达到了非常好的优化效果.该算法为求解复杂优化问题提供了新的思路。

针对现有的工作,我们将对以下问题展开进一步的研究:

(1) 进一步研究算法的效率及收敛性;

(2) 研究针对大规模数据集,寻找新的语义关系算子,从而保证算法的寻优能力。

参 考 文 献

[1] Hackwood S, Beni G. Self-organization of Sensors for Swarm Intelligence[C]//IEEE International conference on Robotics and Automation. Piscataway, NJ: IEEE Press, 1992: 819-829.

[2] Kennedy J, Eberhart R C, Shi Y. Swarm intelligence[M]. Morgan Kaufmann,2001.

[3] Bonabeau E, Dorigo M, Theraulaz G. Swarm intelligence from natural to artificial systems[M]. Oxford University Press Inc, 1999:1-22.

[4] White T. Swarm Intelligence: A Gentle Introduction With Application [EB/OL]. http://www.see.carleton.ea/netmanage/tony/swarm-presentation/index.htm.

[5] Flocking and Collective Movement[M/OL]. http:// www.coro.ealteeh.edu/Courses/EE141/Lecture/WZ/OH_EE141_W2flocking.pdf.

[6] Fulkerson B. Swarm Intelligence-What Is It and Why Is It Interesting? [EB/OL] http://www.coro.ealteeh.edu/Courses/EE150/ Weekl/OH_W1SwarmIntel.Pdf.

[7] 张铃,程军盛. 松散的脑袋—群体智能的数学模型[J]. 模式识别与人工智能, 2003,16(1):1-5.

[8] 葛芬,吴宁. 基于多种技术的 Word 设计文档自动生成平台[J]. 电子科技大学学报, 2007,36(2):263-266.

[9] 艾伟,孙四明,张峰. 基于本体的 Web 文本挖掘与信息检索[J]. 计算机工程, 2010, 36(22):75-80.

[10] Arpirez J C, Corcho O, Fernandez-Lopez M, et al. WebODE: A scalable ontological engineering workbench[C]//GilY, Musen M, Shavlik J. Proc. of the K-CAP200! NewYork, ACM Press, 2001:6-13.

[11] 高尚,杨静宇. 群智能算法及其应用[M]. 水利水电出版社, 2006:1-20.

[12] Chu Kuo-kuang, Lee Chien'a, Tsai Rong-shi. Ontology technology to assist learners' navigation in the concept map learning system[J]. Expert Systems with Applications, 2010,38:11293-11299.

[13] Zhao Chongchong, Wang Jing, Hu Wei, et al. An Ontology-based Semantic Search Model Study[C]//2010 3rd Interational Symposium on Knowledge Acquisition and Modeling, 2010:182-185.

(上接第 179 页)

间通信和连接超时对性能有显著影响,我们认为,在节点较少时,可将该爬虫部署在 LAN 中,具有更好的经济性,当节点较多时,由于 LAN 下重复获取网站增多,资源利用率下降,应采用 WAN 的方式部署,实现大规模的有效应用。在此基础上,通过调节连接超时阈值,可有效地提高爬虫性能,在 LAN 中可以选择较大的阈值,在 WAN 环境中应选择较小的阈值。负荷较低的爬行节点可以通过提高超时阈值,从而抓取更多的网络主机,来提高自己的负荷。负荷较高的爬行节点则可以降低自己的超时阈值,同时可以通过增加爬行节点数量来提高爬虫系统整体下载速度以满足需求。

5 结 语

本文设计了一种新型的分布式网络爬虫 DSpider。它将 Web 资源以主机名进行划分,并且以网络连接阈值为基础进行节点任务的分配与调度。对 DSpider 的系统架构和任务调度策略作了详细的分析,并且作了仿真实验,讨论了 DSpider 部署在 LAN 和 WAN 两种环境下的不同性能表现。

本文设计的 DSpider 仅对部署于 LAN 和 WAN 两种环境下的爬虫性能做了初步研究,对于 Web 文件主题提取、复杂网络环境下部署模式等复杂情况尚未深入探索,有待于今后做更进一步的研究。

参 考 文 献

[1] Gao Qing, Xiao Bo, Lin Zhiqing, et al. A High-precision Forum Crawler Based on Vertical Crawling[C]//Proceedings of the 2009 International Conference on Network Infrastructure and Digital Content,2009.

[2] Vladislav Shkapenyuk, Torsten Suel. Design and Implementation of a High-Performance Distributed Web Crawler[C]//Proceedings of the 18th International Conference on Data Engineering.

[3] Yuan Wan, Hengqing Tong. URL Assignment Algorithm of Crawler in Distributed System Based on Hash[C]//IEEE International Conference on Networking, Sensing and Control,2008.

[4] Su Xuan. The Research, Implement on Technology of Distributed Web Crawler. School of Computer Science and Technology.

[5] Guerriero A,Pasquale C,Ragni F. Java based architecture for Grid Application[C]//Proceeding of VECIMS 2009-IEEE International Conference on Virtual Environments, Human-Computer Interfaces, and Measurement Systems May 11-13, 2009 Hong Kong, China.

[6] Cho j,Garcia-Molina H. Parallel crawlers[C]//Proceedings of the 11th International Word Wide Web Conference, New York, NY, USA, 2001:124-135.

[7] Boldi P, Codenotti B, Santini M, et al. Ubicrawler: A scalable fully distributed web crawler[J]. Software, Practice and Experience, 2002, 34(8):711-726.

[8] Li XM,Yan I-IF,Wang JM. Search Engine:Principle,Technology and System[M]. Beijing:Science Press,2005(in Chinese).

[9] Ye YM,Yu S,Ma FY,et al. On distributed Web crawler:Architecture, algorithms and strategy[J]. Acta Electronic Sinica,2002,30(12A): 2008-2011(in Chinese with English abstract).

[10] Chen Y,Chen Y B. Tree Reconstruction and Bottom-up Evaluation of Tree Pattern Queries[C]//Int. Conf. on Information Science and Applications (ICISA 2010), Seoul, Korea.