# 主动获取式的分布式网络爬虫集群方法研究

董禹龙 杨连贺 马 欣
(天津工业大学计算机科学与软件学院 天津 300387)

**摘 要** 针对当前分布式网络爬虫方法遇到的处理效率、扩展性、可靠性、任务分配和负载平衡等问题,提出了一种主动获取任务式的分布式网络爬虫方法。该方法在子机节点中加入分控模块,评估节点负载及运行状况,并主动向中控节点申请任务队列。在此基础上,结合动态双向优先级任务分配算法,设计了一种具有负载平衡、任务分级分配、节点异常敏捷识别、节点安全退出等特性的分布式网络爬虫模型。实际测试表明,该主动获取式的分布式网络爬虫方法可有效地利用通用平台建立大型分布式爬虫集群。

**关键词** 主动获取,分布式爬虫,负载平衡,爬虫框架,多进程,动态优先级
**中图法分类号** TP301.6 **文献标识码** A

## Study on Active Acquisition of Distributed Web Crawler Cluster

DONG Yu-long YANG Lian-he MA Xin
(School of Computer Science and Software Engineering, Tianjin Polytechnic University, Tianjin 300387, China)

**Abstract** In this paper, in order to solve the processing efficiency, scalability, task allocation and load balance problem existed in the present distributed web crawler method, an active acquisition task distributed web crawler method was proposed, in which a sub-controlled module is added into the sub-node to evaluate the node load and operation status, and apply task queue for the central control node. Based on this method as well as the dynamic dual-directional priority task allocation algorithm, a distributed network crawler model was designed, which has the characteristics of load balance, task hierarchical allocation, abnormal node smart identification and safe exit, etc. The practice test shows that the active acquisition task distributed web crawler method can be used to build large-scale distributed crawler cluster effectively.

**Keywords** Active obtain, Distributed crawler, Load balancing, Crawler framework, Multi process, Dynamic priority

## 1 引言

随着云计算时代的来临,各行各业产生的数据与日俱增,海量的数据不断涌现。IDC 报告显示,预计到 2020 年,全球数据总量将超过 40ZB(相当于 4 万亿 GB)。利用数据科学技术分析出数据中隐藏的关系、模式、趋势,并提供预测性决策支持,是企业保持竞争力的必要手段。与国外相比,我国的信息化程度尚不够,企业的信息存储意识亦不强,这使得现实中数据来源困难。想要得到大量的符合要求的数据往往需要依靠网络爬虫来收集[1],这也使得高效的网络爬虫技术对我国应对目前越来越激烈的市场竞争有着重要的战略意义[2-3]。

近几年来,随着计算机技术的不断发展,网络爬虫技术从最初的单一网络爬虫向分布式并行网络爬虫发展;爬行的内容从静态的、单一的网页向动态的、实时加载的网页发展;爬行的范围从原来的通用网络爬虫向主体的、聚焦的网络爬虫发展;爬虫的目的越来越明确,搜索的范围也越来越精准[4-8]。目前的分布式网络爬虫主要采用主从式的分布方式,即由一个中控节点控制着每一个子机节点的结构,由中控节点对每个子机节点进行任务分配。但随着实际应用中爬行任务数的不断增多,人们发现,中控节点的性能将会成为制约整个系统效率的瓶颈,从而降低整个分布式网络爬虫系统的性能。同时,由于各节点的性能存在差异,长时间运行还会造成节点负载不均衡现象,即个别节点压力过重,而其他节点处于空闲的状态。

为了解决上述问题,本文结合现有的爬虫技术和分布式调度算法的优点,提出一种主动获取式的分布式网络爬虫方法。从分析该方法中各模块的功能与性能出发,结合异步式网络爬虫和分布式网络爬虫的优点,引入了子节点控制模块,分散了中控节点的压力,并解决了子节点的负载均衡问题。通过结合动态的双向优先级任务分配和调度算法[9],实现了各节点的负载平衡,优化了爬行路径,并通过实验证明了这种分布式集群框架在任务分配、负载均衡、节点异常敏捷识别、异常节点安全退出、动态性、扩展性、可靠性等方面都有很好的表现。

## 2 研究框架

本文的分布式网络爬虫使用 Python 语言实现。

### 2.1 系统模型

本文提出的分布式爬虫框架模型如图 1 所示,中控节点运行在配置相对较高的服务器上,负责集群系统的调度,其主要任务包括:收集子机节点返回的任务,并对其进行过滤优化;通过子机评价算法评价子机性能;控制任务爬取频率和优先级判定。子机节点运行在分散的各台主机上,由子机控制模块分别控制子机上的各个爬行队列。

**董禹龙**(1991—),男,硕士生,主要研究方向为数据挖掘及分布式系统,E-mail:dongyulong1991@sina.com。

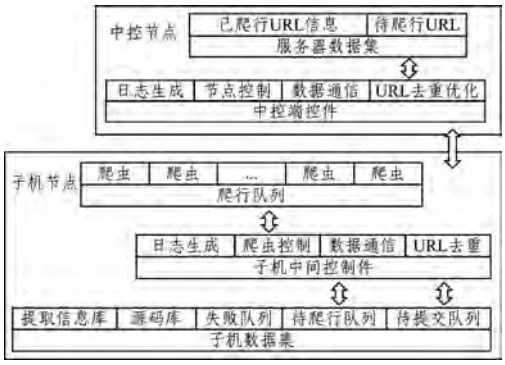



图 1　分布式爬虫框架模型

系统运行流程如下：

1）用户输入种子 URL 文件，由中控节点对任务队列进行优化，等待子机节点连入。

2）子机节点连接中控节点并发送数据请求，中控节点向子机发送任务队列；子机将任务队列存入待爬行队列，并控制爬虫爬取。

3）爬虫从待爬行队列库中取出 URL，并在 Internet 上开始下载，将下载成功的文件保存在本地信息库中，并将网页源码传给信息提取器。

4）提取器处理网页源码。提取器根据用户自定义的提取规则模板提取信息和下一级的 URL，并将提取的信息存于本地，更新已爬行表，并将 URL 提交给过滤模块。在该模块中，根据用户的不同需求，通过自定义提取规则可以实现不同网站、不同信息的多样化提取，在其间加入一个主题判定过滤器，便可实现特定信息的高效提取。

5）URL 过滤器过滤掉不合格的 URL，并根据本地已爬行表信息和待提交队列初步剔除重复的 URL。

6）待提交 URL 暂时保存到子机的待提交队列，当达到目标数量后，提交给中控节点并实施更新。

7）中控节点收到各个子机提交的 URL，进行任务生成后放入中控节点待爬行表，等待子机任务请求。当不再有新的 URL 提交，并且所有子机任务结束时，系统运行结束。系统运行流程如图 2 所示。

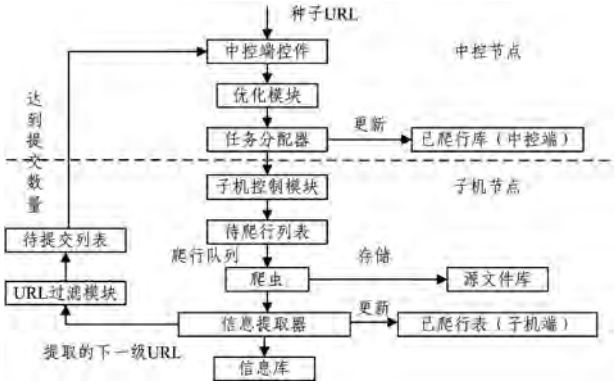
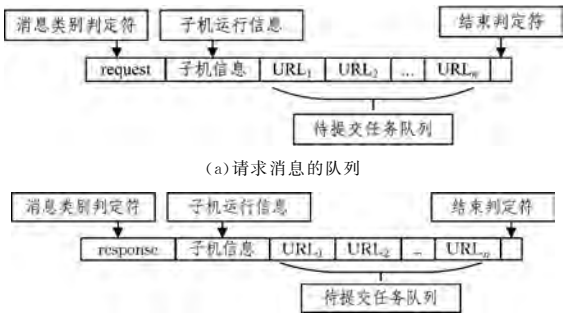



图 2　系统运行流程示意图

## 2.2　节点间通信

### 2.2.1　通信协议

分布式集群系统的通信主要表现为中控节点和各个子机节点间的通信。按照约定好的通信协议，中控节点将子机节点发送的消息队列识别为响应队列或请求队列，两种消息队列如图 3 所示。




(a)请求消息的队列

(b)响应消息队列

图 3　子机发送消息的结构图

其中，消息队列头部为消息类别标识，有两种类型。

1）“request”：中控节点将其识别为请求队列，通过消息队列的子机信息域判断请求子机编号和子机状态，然后在任务库中提取任务队列并发送返回消息。为降低报文开销并减少报文发送次数，将子节点失败返回队列与请求报文同时返回，结尾为约定的消息结束符，用于判断消息是否结束。

2）“response”：中控节点将其识别为响应队列，当子机的待提交任务队列达到可提交数量阈值时，调用子机消息模块发送响应消息队列。

### 2.2.2　动态任务提交机制

在爬行初期或爬行任务较少时，中控节点将任务全部分配到各个子机节点，而子机节点生成的 URL 数量达不到提交阈值时，会造成中控节点无任务分配、子节点无任务提交的假象，从而影响集群系统的整体运行效率。鉴于此，本文在消息队列中设置了一个动态任务返回信号，以实现双向动态任务请求，如图 4 所示。

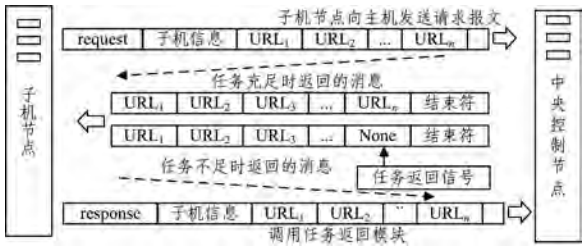



图 4　双向动态任务返回示意图

图 4 表明，当中控节点任务量不足且子机节点请求任务时，中控节点将动态任务返回信号写入消息队列。子机收到中控节点返回的信息并进行消息识别，当识别出返回信息后，主动调用消息返回模块返回待爬行队列，并重新发送请求消息队列。此即“动态任务提交”之含义。

# 3　任务分配与调度算法

## 3.1　主动式任务获取

传统的基于分布式哈希表（Distributed Hash Table，DHT）结构的分布式爬虫任务划分主要由中控节点来实现。其任务分配策略是，指定一台主机作为中控节点，将用户指定的 URL 哈希成与子机节点数相同的 Key 值，并根据 Key 值将 URL 分配给最近的子机节点。因为各子机节点的处理速度和配置等因素不同，可承受的负载等级也不尽相同，所以这样的分配策略将会带来一个很明显的问题——负载不均衡，即有的节点超载运行，而其他节点则空闲[10-11]。这也是许多研究者争相提出改进算法的焦点。Rao 等[12]提出了虚拟服务器的方法，即将实际节点虚拟成多个互相独立的虚拟节点，节点间进行负载的迁移；David 等[13]使用多个哈希函数虚拟

节点的位置，以便根据负载情况进行动态变化，尽可能地使ID空间均匀分布；Rieche 等[14]通过类似热扩散的物理原理划分 ID 区间，在每个区间设置冗余的多个节点，将负载在邻近节点间进行转移，以达到负载平衡的目的。

不同于 DHT 结构的分布式网络爬虫，本文提出的分布式网络爬虫框架在子机节点中加入了子机控制模块，将传统的由子机节点被动地接受中控节点分配的任务变成由子机控制模块根据子机负载压力系数主动地获取任务。

### 3.2 负载压力系数

本文提出的负载压力系数由当前节点的平均处理速度、任务组剩余量和最大压力时间决定。

子机节点从中控节点接收到 URL 队列后形成本地任务队列，子机控制模块将当前任务队列的剩余任务量和任务压力阈值进行对比。其中，任务压力阈值由子机运行的平均速度和预设压力时间决定，当任务量达不到任务压力阈值时，子机控制模块向中控节点发送任务请求。

综上所述，定义子机节点的负载压力系数为：

$$\lambda = W - VT \tag{1}$$

其中，$W$ 随时间变化逐渐减小，当本地任务队列的任务数减小到一定值时，$\lambda$ 变为负值，表示此子机节点处于轻负载状态，此时子机控制模块向中控节点发送请求命令。

### 3.3 任务分配策略

在集群式的分布式系统中，中控节点通过任务调度算法将任务分配到各个节点上进行爬取。URL 的页面价值各不相同，拥有高价值、高时效性的 URL 任务需要被优先爬取，以保证高价值任务的优先发现和优先处理；同时，对低时效性的网页则可以延后爬取。优先级分类的原则是：将不同重要性的 URL 进行分级，高优先级的任务优先分配执行。考虑到不同节点处理能力的差异和各节点请求任务时间的间隔，参考基于动态双向优先级的任务分配与调度算法[9]，本文通过综合考虑子机和任务的动态优先级对任务进行分配。

### 3.4 子机优先级算法

在中控系统中，每台子机的优先级用一个四元组 $Sub_i(v_i,\rho_i,conf_i,\omega_i)$ 表示。其中，$v_i(i=1,2,\cdots,n)$ 表示子机的平均执行速率，代表一段时间内子机执行任务的性能；$\rho_i$ 表示爬取 URL 的成功率，反映了子机网络传输和系统执行等综合因素的影响；$conf_i$ 表示子机实际执行速率的置信度，用于评价约定时间；$\omega_i$ 表示子机权重，为预设值，由于各个子节点完成爬取任务后会将爬取结果分布式地存储在本地，因此 $\omega_i$ 高的子机优先保存高价值的任务。

子机 $Sub_i$ 的优先级 $Sub\_p_i$ 的计算公式为：

$$Sub\_p_i = \{v_i \times (1-\beta) + conf_i \times \beta\} \times \omega_i \times \rho_i \tag{2}$$

其中，$\beta$ 为协调因子，负责调和 $v_i$ 和 $conf_i$ 对 $Sub\_p_i$ 的影响，避免任务过度集中或长期空闲；$\rho_i$ 为子机平均执行成功率，保证了高成功率的子机优先分配高价值的任务。

应当强调的是，引入爬取成功率 $\rho_i$，可以帮助中控节点判断子机当前是否发生异常。当 $\rho_i$ 趋近于 0 时，表明子机节点出现异常，此时将不再把任务分配给相应的子机节点。同时，子节点的失败任务以消息队列的形式返回给中控节点重新进行分配，在保障任务完整的前提下使异常子机安全退出。

式(2)中，$v_i$ 的计算公式为：

$$v_i = \begin{cases} \dfrac{n}{\sum_{i=1}^{n} T_i}, & n \neq 0 \\ v_{pre}, & n = 0 \end{cases} \tag{3}$$

其中，$n$ 为相邻两次请求间子机执行的任务数。$v_i$ 的定义是，当子机每次发送报文请求时，更新平均速率 $v_i$ 并重新开始计数。$n$ 为 0 时，将 $v_i$ 赋值为初始先验速率 $v_{pre}$。$T_i$ 为子机节点实际执行每条任务的时间。

式(2)中，$conf_i$ 的计算公式为：

$$conf_i = \frac{\bar{v}_i}{\sqrt{\frac{1}{n}\sum_{j=1}^{n}(v_{ij}-\bar{v}_i)^2}} \tag{4}$$

其中，$\bar{v}_i$ 为子机 $Sub_i$ 前 $n$ 次提交的速率 $v_{ij}(j=1,2,\cdots,n)$ 的平均值。$conf_i$ 的值越大，表示子机在当前时间段的处理速度的变化幅度越小，即下次提交的 $v_i$ 值的可信度越高；反之，$conf_i$ 值越小，表明子机当前时间段的处理速度的变化幅度越大，即提交的 $v_i$ 值的可信度越低。

### 3.5 任务分级分配算法

在实际的任务分配过程中，由于各个子机节点效率的差异和请求时间间隔的不同，简单地根据子机个数将较高优先级的 URL 任务按桶分配，可能存在由于子机任务滞留而造成的高优先级任务在子机中实际被延后执行的问题。

假设存在子机节点组 $Sub_i=\{Sub_1,Sub_2,\cdots,Sub_n\}$，$Sub_1$ 请求分配任务数为 $S$，如果将中控节点任务列表中优先级最高的任务集 $H=\{n_1,n_2,\cdots,n_s\}$ 全部分配给 $Sub_1$，下一时刻 $Sub_2$ 请求任务，则任务 $n_s$ 在子机 $Sub_1$ 中的执行时间显然晚于 $n_{s+1}$。

为了防止这一问题，将任务队列按优先级分为两个区域，即 high 区和 low 区。根据子节点优先级在全部节点中的比重，将请求任务集 $H$ 划分为 $H_{high}$ 和 $H_{low}$。其中，$H_{high}$ 表示在 high 区取出的任务集，$H_{low}$ 表示在 Low 区取出的任务集。子节点优先级越高，分得的 $H_{high}$ 比例越大。$H_{high}$ 和 $H_{low}$ 的计算公式为：

$$H_{high} = \frac{Sub\_p}{\frac{1}{n}\sum_{i=1}^{n} Sub\_p_i} \times S_{high} \tag{5}$$

$$H_{low} = H - H_{high} \tag{6}$$

其中，$S_{high}$ 为中控节点任务列表中 high 区的任务总数，$Sub\_p$ 为当前请求节点的优先级。当 high 区分配完毕或者任务数不足时，在 low 区提升优先级最高的 URL 添加到 high 区。high 区的任务优先级为静态，low 区的任务优先级随着子机节点不断地提交任务而动态地更新，如图 5 所示。

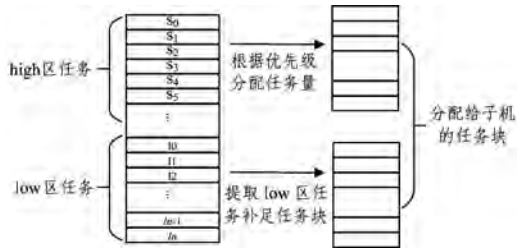



图 5 任务分级分配

## 4 实验与仿真

本文提及的爬虫系统用 Python 语言编程实现，由运行在大型计算机集群上的一台高配置中控节点和 6 个子机节点测试，各节点的配置信息如表 1 所列。为了更加全面地介绍本体系结构的爬虫性能，下面从单机爬取和集群系统爬取两方面进行性能测试。

表 1　爬虫系统测试环境的参数说明

| 地址编号 | 任务 | CPU 数 | 内存/GB | 网络配置/Gbps |
|---|---|---|---|---|
| xx.67.107.122 | 中控节点 | 16 | 16 | 1 |
| xx.67.107.94 | 子机节点 | 8 | 8 | 1 |
| xx.67.107.95 | 子机节点 | 8 | 8 | 1 |
| xx.67.107.118 | 子机节点 | 8 | 8 | 1 |
| xx.67.107.119 | 子机节点 | 8 | 8 | 1 |
| xx.67.107.120 | 子机节点 | 8 | 8 | 1 |
| xx.67.107.121 | 子机节点 | 8 | 8 | 1 |

### 4.1　单机运行性能测试

本实验将测试子机中并行爬虫的数量对节点效率的影响。采样周期为 5 min，采样 5 次，初始 URL 为 1 条，子机节点的 CPU 使用情况和爬取 URL 的条数如图 6 所示。

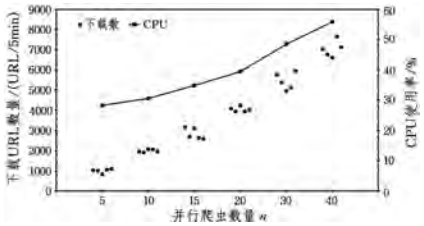



图 6　系统效率和资源的使用情况

由图 6 可知，通过增加子机的并行爬虫数量，可以加快爬行速率。当并行爬虫数量少于 20 时，爬取效率随爬虫数量呈线性增长；当爬虫数量多于 20 时，爬取效率的增势趋缓（注意，横轴非线性）。在实际的应用中，可根据子机节点的性能对并行节点数进行适当调节，以最大化节点效率或者优化 CPU 的使用情况。

### 4.2　集群系统的性能测试

分布式爬虫集群系统的下载能力主要受子机节点数和并行在各子机节点上的爬虫数量的影响。本文提出的分布式网络爬虫方法具有高扩展特性，随着分布节点数量的增加，下载速度和页面信息处理速度呈线性增长。

#### 4.2.1　扩展性测试

设置每个子机节点并行运行相同数量的爬虫，系统的采样周期为 5 min，每隔 5 个采样周期加入一个子机节点。在任务量充足的条件下，采样数据如图 7 所示。

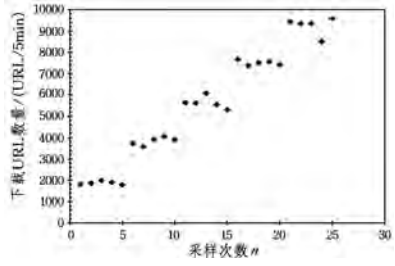



图 7　扩展性测试采样

在图 7 中，每当有新的子机节点加入时，系统 URL 的下载速度都会有明显的提升；而且，系统的平均处理速度随子机节点数呈线性增长，如图 8 所示。

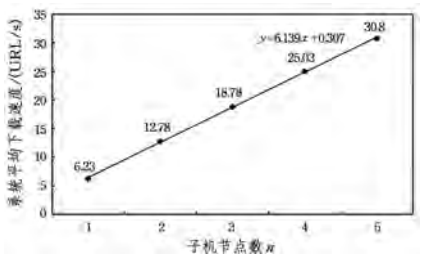



图 8　系统扩展图

由此可见，本文提出的主动获取式分布式爬虫系统有着良好的扩展性。在实际应用中，可以根据需要增加子机节点数，以提高爬取效率。

#### 4.2.2　负载平衡测试

设置实验条件，使 1 个中控节点和 6 个子机节点并行运行不同数量的爬虫。统一设置负载压力时间为 120 s。首先测试在未采用负载平衡时各节点的负载倍率，然后将其与采用了负载平衡策略的各节点的负载情况进行对比，各节点的信息如表 2 所列。

表 2　两组样本中各节点的信息

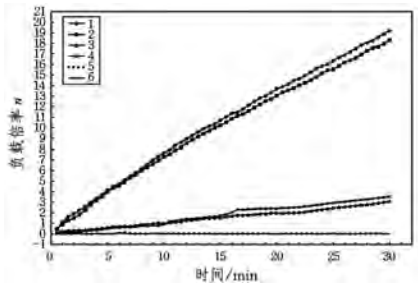
| 节点序号 | 并行爬虫数 | 未采用负载平衡时运行效率 | 采用负载平衡时运行效率 | 爬行总数（未采用/采用） | 压力栈大小（未采用/采用） |
|---|---|---|---|---|---|
| 1 | 5 | 3.42 | 3.56 | 6280/6400 | 410/427 |
| 2 | 5 | 3.54 | 3.34 | 6351/6003 | 425/401 |
| 3 | 10 | 6.97 | 7.22 | 12551/12988 | 836/866 |
| 4 | 10 | 6.63 | 6.92 | 12041/12464 | 796/830 |
| 5 | 15 | 8.30 | 10.41 | 14931/18739 | 996/1249 |
| 6 | 15 | 8.43 | 10.19 | 15182/18304 | 1012/1220 |

通过表 2 可以看出，对于未采用负载平衡策略的样本，当并行爬虫数达到 15 时，其运行效率已明显低于采用负载平衡策略的效率，这是由于任务数不足时个别爬虫空闲造成的。

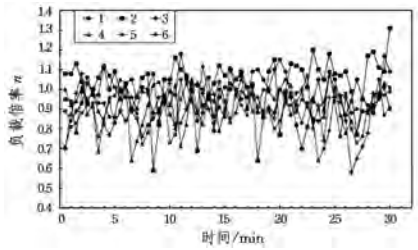
本文通过样本的负载倍率更直观地表示两组样本的负载状况。负载倍率的公式如下：

负载倍率＝待爬行任务数 / 任务压力阈值

可以看出，当负载倍率为 1 时状态最佳，其意义是，任务队列中的待爬行 URL 数量刚好在压力时间内爬行完毕。从图 9(a)中可以看出，如果子机节点的爬行速度低于任务分配的速度，子机任务量饱和，多余的任务将在子机处堆积，随着时间的推移，将有大量数据得不到及时处理；相反，当子机节点的运行速度超过任务分配速度时，则会造成子机节点长时间处于空载状态，大大降低了子机的运行效率。




(a)未使用负载平衡策略




(b)使用负载平衡策略

图 9　节点负载平衡的对比

而在图 9(b)中，采用了主动获取式负载平衡策略后，子机将自行判断负载压力系数的变化。可以看出，无论子机的运行速率如何变化，其负载倍率始终在 1 左右徘徊，既不会出现任务堆积，也不会出现空载状况。

**结束语**　本文提出了一种主动获取式的分布式网络爬虫框架，通过实验证实该框架具有异常节点识别、节点安全退

出、数据动态更新的功能，并具有高扩展性和高可靠性的特点。同时，本文的负载平衡算法、动态双优先级任务分配算法在实际中也有着广泛的应用前景。下一步将对本文提出的爬虫框架和算法做进一步优化，以期在更多的领域中获得应用。

## 参 考 文 献


[1] ZHOU D M. Survey of High-performance Web Crawler[J]. Computer Science,2009,36(8):26-29.

[2] 周孝锞. 基于网络爬虫和改进的 LCS 算法的网站更新监测[J]. 计算机应用与软件,2017,34(1):222-229.

[3] 周德懋,李舟军. 高性能网络爬虫:研究综述[J]. 计算机科学,2009,36(8):26-29.

[4] BRIN S,PAGE L. Reprint of:The anatomy of a large-scale hypertextual web search engine[J]. Computer Networks,2012,56(18):3825-3833.

[5] MADAAN R,SHARMA A K,DIXIT A. A novel architecture for a blog crawler[C]// IEEE International Conference on Parallel Distributed and Grid Computing. IEEE,2012:452-456.

[6] THAU B,OWEN L,KRISHNAMURTHY C S. Distributed Web Crawling over DHTs:UC Berkeley Technical Report UCB,CSD-4-1305[R]. 2004.

[7] ZHU K,XU Z,WANG X,et al. A Full Distributed Web Crawler Based on Structured Network[C]// Asia Information Retrieval Conference on Information Retrieval Technology. Springer-Verlag,2008:478-483.

[8] 刘爽,姜春祥,张伟哲,等. 基于 GNP 算法的分布式爬虫调度策略[J]. 计算机应用研究,2010,27(2):446-449.

[9] 龚跃,张真真,黄小珂,等. 基于动态双向优先级的任务分配与调度算法[J]. 计算机应用,2009,29(4):1131-1134.

[10] 黄志敏,曾学文,陈君. 一种基于 Kademlia 的全分布式爬虫集群方法[J]. 计算机科学,2014,41(3):124-128.

[11] 陶耀东,向中希. 基于改进 Kademlia 协议的分布式爬虫[J]. 计算机系统应用,2016,25(4):156-161.

[12] RAO A,LAKSHMINARAYANAN K,SURANA S,et al. Load Balancing in STRUCTURED P2P Systems[M]// Peer-to-Peer Systems II. Springer Berlin Heidelberg,2003:68-79.

[13] KARGER D R,RUHL M. Simple Efficient Load-Balancing Algorithms for Peer-to-Peer Systems[M]// Peer-to-Peer Systems III. Springer Berlin Heidelberg,2005:131-140.

[14] RIECHE S,PETRAK L,WEHRLE K. A thermal-dissipation-based approach for balancing data load in distributed hash tables [C]// IEEE International Conference on Local Computer Networks. IEEE Computer Society,2004:15-23.


(上接第 405 页)

**结束语** 本文提出了一种基于不确定理论和情感分析的个性化推荐算法，该算法从在线评论中挖掘用户的情感倾向来建立情感分析模型；基于该模型设计了个性化推荐算法。在推荐精度及数据稀疏问题上，所提算法比对比算法更具优势。目前，虽然情感分析技术与的很多方法结合并应用于推荐算法的研究中，但情感分析技术结合不确定理论应用到推荐算法中仍处在发展阶段，需要开展大量的研究与实际应用来评价该算法的理论意义与应用价值。

## 参 考 文 献


[1] RESNICK P,VARIAN H R. Recommender systems [J]. Communications of the Acm,1997,40(3):56-58.

[2] DE CAMPOS L M,FERNÁNDEZ-LUNA J M,HUETE J F. A collaborative recommender system based on probabilistic inference from fuzzy observations[J]. Fuzzy Sets and Systems,2008,159(12):1554-1576.

[3] MURTHI B,SARKAR S. The role of the management sciences in research on personalization[J]. Management Science,2003,49(10):1344-1362.

[4] CHENG L C,WANG H A. A fuzzy recommender system based on the integration of subjective preferences and objective information[J]. Applied Soft Computing,2014,18(C):290-301.

[5] SON L H. HU-FCF:A hybrid user-based fuzzy collaborative filtering method in Recommender Systems [J]. Expert Systems with Applications,2014,41(15):6861-6870.

[6] DRAGONI M,TETTAMANZI A G B. Using fuzzy logic for multi-domain sentiment analysis[C]// International Conference on Posters & Demonstrations Track. 2014.

[7] HAQUE M A. Sentiment Analysis by Using Fuzzy Logic [J]. Computer Science,2014,4(1):33-48.

[8] LOIA V,SENATORE S. A fuzzy-oriented sentic analysis to capture the human emotion in Web-based content[J]. Knowledge-Based Systems,2014,58(1):75-85.

[9] WANG B,HUANG Y,WU X,et al. A Fuzzy Computing Model for Identifying Polarity of Chinese Sentiment Words[J]. Comput Intell Neurosci,2015,2015(4):1-13.

[10] CHEN L,CHEN G,WANG F. Recommender systems based on user reviews:the state of the art [J]. User Modeling and User-Adapted Interaction,2015,25(2):99-154.

[11] CHEN L,WANG F. Preference-based clustering reviews for augmenting e-commerce recommendation [J]. Knowledge-Based Systems,2013,50(C):44-59.

[12] FU G,WANG X. Chinese sentence-level sentiment classification based on fuzzy sets[C]// 23rd International Conference on Computational Linguistics: Posters. Association for Computational Linguistics,2010:312-319.

[13] XIANGHUA F,GUO L,YANYAN G,et al. Multi-aspect sentiment analysis for Chinese online social reviews based on topic modeling and HowNet lexicon[J]. Knowledge-Based Systems,2013,37:186-195.

[14] LIU B. Uncertainty Theory[M]. Spring Publishing Company,2007.

[15] LIU B. A branch of mathematics for modeling human uncertainty [M]. Berlin:Springer-Verlag. 2011.

[16] DONG Z,DONG Q,HAO C. HowNet and the Computation of Meaning[M]. World Scientific Publishing Co. Inc. ,2006.

[17] TURNEY P D. Thumbs up or thumbs down?:semantic orientation applied to unsupervised classification of reviews[C]// Proceedings of the 40th annual meeting on association for computational linguistics. Association for Computational Linguistics,2002.

[18] LIU B. Uncertain logic for modeling human language [J]. Journal of Uncertain Systems,2011,5(1):3-20.

[19] SARWAR B,KARYPIS G,KONSTAN J,et al. Item-based collaborative filtering recommendation algorithms[C]// 10th International Conference on World Wide Web. ACM,2001.

[20] WANG X,GAO Z,GUO H. Delphi method for estimating uncertainty distributions[J]. International Information Institute (Tokyo),2012,15(2):449-459.