基于云平台的分布式高性能网络爬虫的研究与设计

石恩名，肖晓军，卢宇
（广州优亿信息科技有限公司，广东 广州 510630）

摘　要：随着大数据时代的到来，数据成为最宝贵的资源，而网络爬虫技术作为外部数据采集的重要手段，已然成为数据分析的标配。介绍了一种高性能、灵活和便捷的基于云平台的爬虫架构设计和实现。从爬虫的整体架构、分布式设计以及各模块的设计等角度进行了详细的阐述。爬虫各模块用 Docker 封装，Kubernetes 做集群的资源调度和管理，在性能优化上采用了 MD5 去重树算法、DNS 优化和异步 I/O 等多种策略组合的形式。实验表明，对比未优化的方案，爬虫在性能上具有较明显的优势。

关键词：分布式系统架构；网络爬虫；Docker；高性能
中图分类号：TP393　　**文献标识码**：A
doi: 10.11959/j.issn.1000−0801.2017234

Research and design of distributed high-performance network reptiles based on cloud platform

SHI Enming, XIAO Xiaojun, LU Yu
Guangzhou Useease Information Technology Co., Ltd., Guangzhou 510630, China

Abstract: With the arrival of large data age, data has become the most valuable resource. And web crawler technology as an important means of external data collection, has become a standard tool for data analysis. A high-performance, convenient cloud-based crawler architecture design was introduced. The overall structure of the reptile to the distributed design and the design of the sub-module was described in detail. Each module of the crawler was encapsulated in Docker, and Kubernetes was used as the resource scheduling and management of the cluster. In the performance of optimization, the MD5 reset tree algorithm, DNS optimization and asynchronous I/O were adopted. Experimental results show that the performance of crawler has obvious advantages compared with the UN optimized scheme.

Key words: distributed system architecture, web crawler, Docker, high-performance

1　引言

大数据时代，数据是最丰富也是最宝贵的，网络信息量快速增长，面对这样海量的数据，如何快速、精确和低成本地收集到所需数据是目前企业所关注的热点。网络爬虫作为一种重要的数据采集手段，既是各类应用的外部数据来源（如内容聚合类应用：今日头条、百度新闻等），也是搜索引擎的重要组成部分。特别随着数据分析技术逐渐在各类互联网企业的应用，加上网上应用开放 API 的缺乏，作为外部数据主要来源的网络爬虫技术已经逐渐成为数据分析类应用产品的标

收稿日期：2017−04−18；修回日期：2017−07−27

配。因此不管是在学术界还是工业界, 对它的研究和改良从未间断过。

一个好的爬虫设计可以从以下几点来评价。

- 高性能。高性能是指爬虫系统的伸缩性、爬取速率，是爬虫架构的最重要的评价指标之一，特别是作为搜索引擎核心部分的通用爬虫，采集性能直接关系到其优劣程度。
- 灵活性。灵活性是指爬虫的策略制定可以灵活改变，爬虫可以根据特定需求进行定制化的采集，如采集页面指定内容、是否渲染页面、是否运行采集等，建立具有个性化的采集任务。
- 便捷性。便捷性是指在构建定制化采集任务时的便捷易用程度，如对于定制化爬虫或者主题爬虫，研究人员是否可以快速构建。

但是现有的爬虫系统大多指针对其中一方面或两方面进行优化，比如一个高性能的通用爬虫，其面对特定的任务时，无法做到定制化策略，缺乏一定的灵活性；很多具备灵活性采集策略的爬虫系统，其在性能优化方面有所欠缺。因此，针对以上 3 个指标，本文提出了一种基于云平台的分布式网络爬虫架构，寻求构建一个高性能、灵活性和便捷性的网络爬虫系统。

2　系统框架设计

2.1　整体架构

爬虫系统的整体框架主要包括四大模块：调度模块、采集模块、解析模块、存储模块，如图 1 所示。爬虫系统接收一个初始配置信息如起始 URL，交递给调度模型进行处理，调度模型根据调度策略的设置将调度请求发送给采集模块，采集模块响应请求，通过 HTTP 下载网页信息，并将网页信息传递给页面解析模块，解析模型调用解析规则对页面进行特定解析，并将数据存入数据库，同时将解析的 URL 传回调度器。

2.2　云平台设计

本系统的云平台底层架构采用 Docker 容器集群代替虚拟机进行构建。Docker 是一个基于轻量级虚拟化技术的容器引擎项目，其采用 LXC（新版的基于 Libcontainter）基于内核的虚拟化技术隔离进程、资源和网络，相比传统的虚拟机技术具备更高的性能和效率，同时具有轻量化、快速启动等优点。

本系统的分布式架构构建在 Kubernetes 集群上，Kubernetes 是一个管理跨主机容器化应用的系统，实现了包括部署、运维和应用弹性伸缩在内的一系列基础功能[1]。爬虫各模块为一个 Pod 容器组，集群通过 Replication Controller（副本控制器）创建、维护和弹性伸缩 Pod 集，爬虫各模块的通信通过 Kafka 分布式消息队列进行，如图 2 所示。

3　调度模块设计

调度模块主要包括了调度器和调度策略两大组件，调度器是调度模块的核心，主要负责 URL

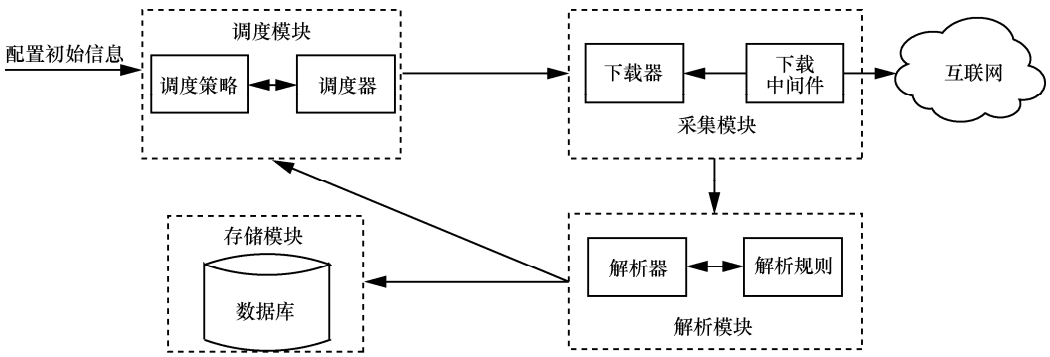


图 1　系统整体框架

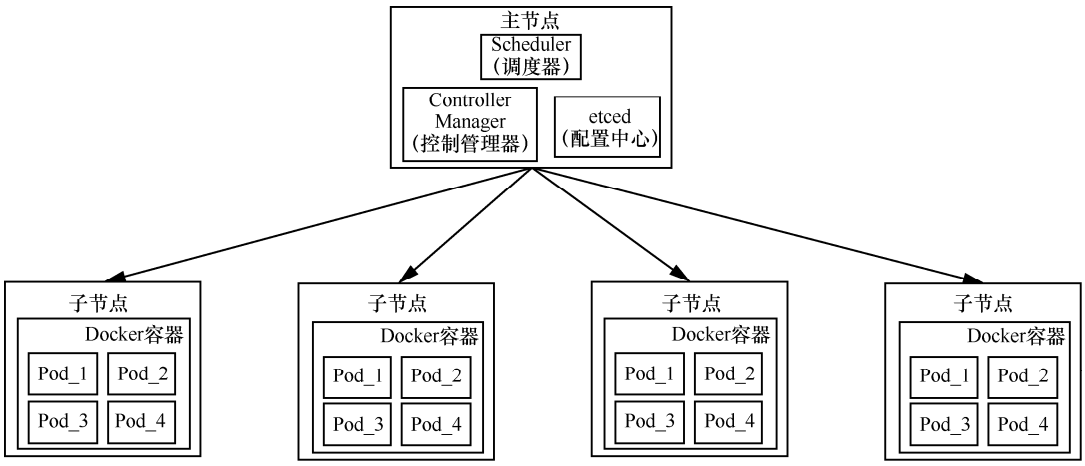


图 2　Kubernetes 分布式框架

的调度和去重等工作，调度策略组件主要为调度器提供策略规则，解释 URL 根据何种方式进行调度和更新，调度策略组件可以根据初始配置进行修改，从而实现不同任务采用不同调度策略。调度模型是爬虫系统性能优化的关键点之一，分布式爬虫任务调度模型核心包括两个部分：URL 的分发服务和去重。

3.1　爬虫任务调度策略

对于爬虫系统来说，大量重复的链接将给搜索引擎带来巨大的负面影响，不但会造成索引存储的内存负担，同时将大大增加索引检索的时间消耗。因此，去重算法的选择是爬虫性能优化的关键。传统的去重算法包括散列算法、布隆过滤器[2]及其各种变形[3,4]，散列算法所需的存储空间大，检索速度慢，但错判率低，可对指定 URL 进行删除；布隆过滤器所需存储空间小，检索速度快，但具有一定几率的错判，其只能判断冲突，不能对指定 URL 进行删除。

由于本系统在设计之初既要兼顾性能，又要兼顾其灵活性和错判率，去重算法上采用严磊提出的基于 MD5 去重树算法[5]进行去重。基于 MD5 去重树是通过将 URL 进行 MD5 压缩，然后构建树结构存储 MD5 值，如图 3 所示。由于其结合了散列算法和树结构的特征，其即具有普通散列算法的低错判率（16 位 MD5 碰撞概率为 $\frac{1}{2^{64}}$），又具有树结构的空间占用小、检索速度快等特点，基于 MD5 去重树在性能和错判率上取得较好的平衡，同时可以满足对指定 URL 的删除功能。

3.2　分布式分发策略

Google 早在 1998 年就提出了 mater-slave（主从）分布式爬虫模型[6]提供 URL 的分发服务。为了保证抓取服务器间的负载均衡，Ubicrawler 提出一种一致性散列的方法来分配任务，抓取服务器负责散列环的一个片段 URL 下载，当某一台服务

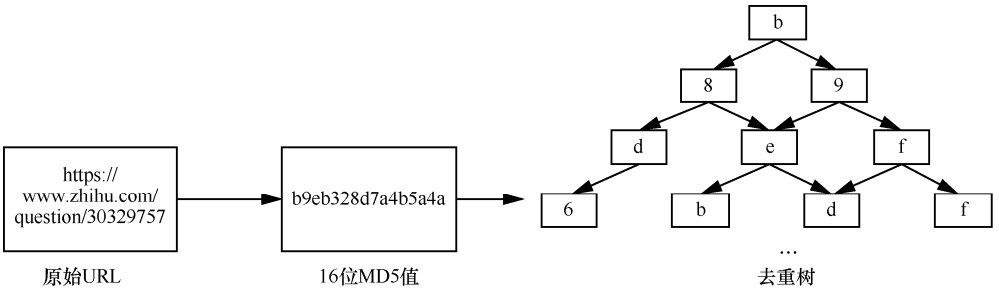


图 3　基于 MD5 去重树算法

器发生异常时，它负责将地址片段上的任务由沿顺时针方向寻找到的第一台服务器处理。一致性散列算法满足了分布式系统中的平衡性、单调性、分散性和负载均衡性。本系统采用 MD5 去重树做 URL 索引，因此在做 URL 服务分发的时候可以直接将去重的 MD5 的值构建一致性散列值，从而将去重和分发结合起来。

4　采集模块设计

采集模块负责抓取网页，是整个系统的基本与关键部分，直接影响爬行效果。采集模块分为两个子模块，一个是下载器，一个是下载中间件。下载器主要是发送下载请求，下载中间件是指对访问网页的请求做加工和处理。

4.1　下载器

下载器主要通过 HTTP 与 Web 服务器进行通信。采用 socket 方式下载的网络编程模型主要有同步 I/O、非阻塞 I/O 和异步 I/O。本系统下载器采用 Tornado Web Server 开发，支持多线程异步 I/O，每个线程的 I/O 连接数维持在 100 个。

对于利用 AJAX 技术动态生产数据的网页，下载器将请求发给 PhantomJS 进行渲染，在 PhantomJS 完成网页 JavaScript 脚本的解析、DOM 树的重构以及浏览器事件触发的模拟，最后 PhantomJS 将渲染后的页面返回。

4.2　下载中间件

下载中间件是利用钩子技术对下载器进行截获，从而构建新的下载请求，本系统的下载中间件包括 DNS 优化器、IP 代理模块等。

（1）DNS 优化器

由于 URL 中域名的大量重复使用，为了提高 DNS 解析效率，本系统引入了 DNS 本地缓存模块，并对 DNS 本地缓存检索、添加、删除进行优化。本系统为 DNS 缓存构建了一个域名散列树主表和一个域名—IP 地址映射表，首先通过散列树将域名的散列值压缩成树结构，然后通过散列值指向域名—IP 地址映射单元，冲突域中按照域名的权重值进行排序，可以定期对 IP 地址做更新。

（2）IP 地址代理模块

IP 地址代理模块主要由 3 部分组成：IP 地址采集器、IP 地址池管理模块、IP 地址代理池。IP 地址采集器负责采集代理网站上的 IP 地址，然后对其进行测试和校验，将可用的 IP 地址、响应时间和运营商等信息存入 IP 地址代理池中。IP 地址池管理模块是 IP 地址代理模块的核心，主要负责 IP 地址调度和 IP 地址校验工作。为了满足高并发的采集需求，本文设计了一种基于域名的 IP 地址代理调度模型。首先将 IP 地址池中的 IP 地址按照响应时间、运营商进行排序，然后首位相连构建成一个环形链表，同一域名下的代理调度从环形链表头部开始，逐级循环下去，这样既可以保证优质 IP 地址被优先使用，同时也保证了同一域名下代理 IP 地址尽可能分散，从而实现了 host 控制的功能。IP 地址池管理模块除了负责 IP 地址的调度工作，还负责 IP 地址的校验更新，IP 地址池管理模块会定时对 IP 地址代理池中的 IP 地址进行校验，并对响应时间进行更新，将不能使用的 IP 地址进行剔除。

5　解析模块

解析模块主要负责内容分析和链接抽取。网页中包含各种各样不同格式的信息，比如文本信息、图片、视频，解析模块就是负责将这些信息抽取出来。解析模块有两个子模块：通用解析子模块和规则解析子模块。

5.1　通用解析器子模块

此模块是指对一些通用的采集、解析任务进行页面解析，通用解析模块内容包括网页正文、图片、视频等，从常用的文本格式如 PDF、txt 中抽取文本信息。正文抽取技术主要采用了朱泽德等人[7]提出的文本密度模型，按行计算网页文本密度，根据邻近行的内容连续性运用高斯平滑获取文本密度，最后利用最大子序列分割的方法抽取正文内容。对于

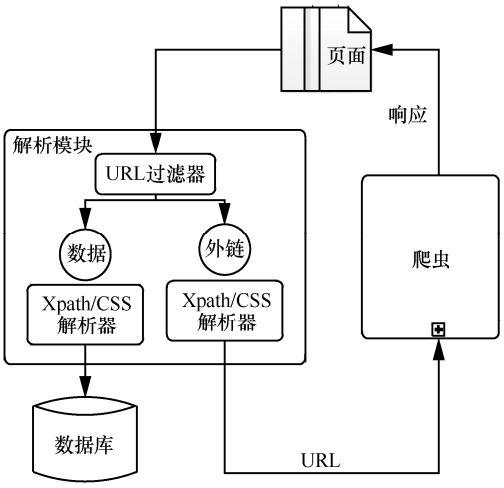
视频和图像等文件，一般是通过链接的锚文本和相关的文件注释锁定解析目标位置。

5.2 规则解析子模块

此模块负责网页分析和信息抽取，根据解析规则对页面进行解析和抽取。解析规则由用户进行设定，图 4（a）是本系统的解析文档示意。解析规则文档主要包括了三大部分：URL-pattern、数据解析、外链解析。URL-pattern 主要是提供过滤器的作用，判断哪些 URL 使用该模板，URL-pattern 支持正则表达式进行匹配。通过 URL-pattern 匹配的页面会被交给数据解析模块，数据解析模块构建 DOM 树，通过 CSS 语法或 Xpath 语法抽取页面内容，根据字段描述信息将解析所得信息存入数据库中，外链解析从页面解析出 URL，通过消息队列将 URL 传回调度器，如图 4（b）所示。值得注意的是，在设计之初考虑到数据解析的灵活性问题，数据解析子模块不仅可以采用 CSS 语法或 Xpath 语法进行解析，同时支持正则表达式解析和特定脚本语言进行解析，如 Python、JavaScript。规则解析可视化界面如图 5 所示。

```
<?xml version="1.0" encoding="UTF-8"?>

<websites>
  <website domain="" host="">

    <url-pattern>url-pattern-1</url-pattern>

    <data-object>
      <field name="title" data-type="string"occur="mandatory"description="">
        <parse>
          <xpath-expression>parse xpath-expression</xpath-expression>
          <str-range start="" end="" />
          <regular-match>abc</regular-match>
          <script-name>function_name<script-name>
        </parse>
      </field>
    </data-object>

    <outlinks>
      <entity occur="mandatory" trace="true" normalize="none" update-interval="-1">
        <parse>
          <xpath-expression>//DIV[@id='pageDivTop']/A</xpath-expression>
          <regular-match>abc</regular-match>
        </parse>
      </entity>
    </outlinks>

  </website>
</websites>
```

（a）解析文档模板示意



（b）规则解析模块流程

图 4　规则解析模块

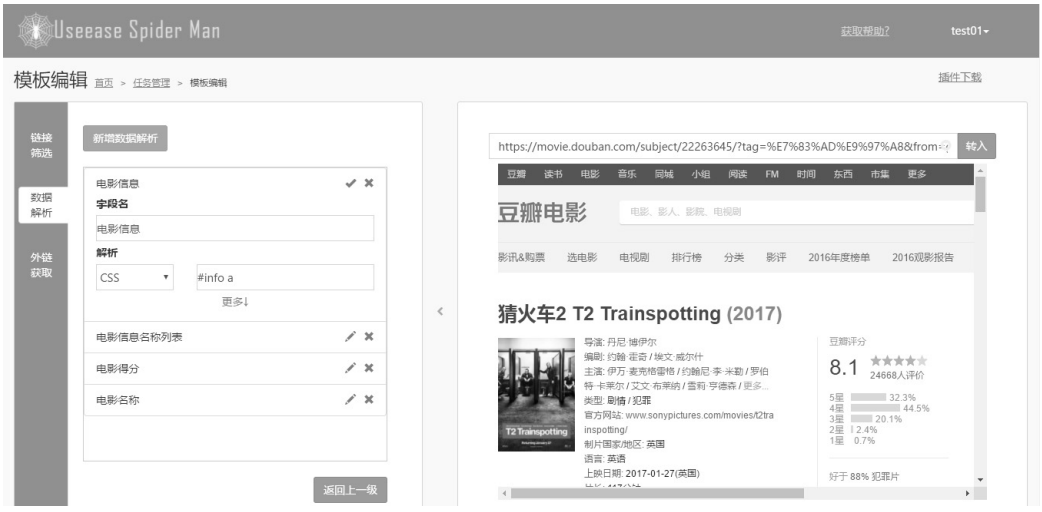


图 5　规则解析可视化界面

6　实验结果与分析

本文以 hao123 网站作为起始 URL，测试该系统爬虫在各方面的性能指标。测试环境为 3 台 CPU24 核，内存 32 GB 的机器，运行环境为 CentOS 6.5，集群共运行 6 h，采集速率如图 6 所示。3 台机器 6 h 共采集数据 500 多万个页面，平均采集速率为 14 099 页/min，每台机器的平均采集速率在 4 699 页/min。

由于整个系统采用了多种优化方案对爬虫的性能进行优化，为了对比各种性能的优化的效果，采用控制变量方法对各指标优化前后做对比，其结果如图 7 所示。

从图 7 可以看出，Hash 去重树算法相对于普通的 Hash 表去重的方法具有非常大的提升，提升了近 30 倍，同时相对于布隆过滤器在性能上并没有受到显著的影响。与未通过 DNS 优化的爬虫相比，性能上也得到了一定的提升。

7　结束语

本文介绍了一种高性能的分布式云平台爬虫框架，底层分布式架构利用 Kubernetes 实现了基于 Docker 的跨主机容器集群自动伸缩，并利用集群对资源进行监控和分配。为构建一个高性能的爬虫，本文采用了 MD5 去重树对 URL 做去重，同时利用 MD5 树所得到的散列值构建一致性散列作为 URL 分发算法实现负载均衡。爬虫的下载器采用异步 I/O 请求，并对 DNS 缓存进行了优化，

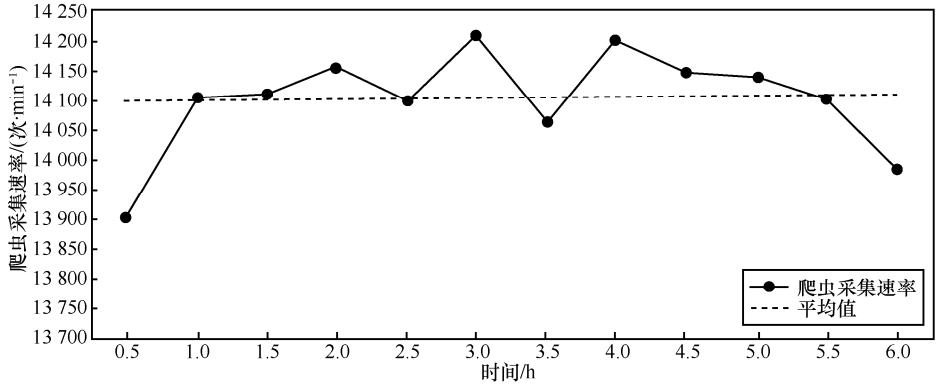


图 6　爬虫系统 6 h 采集速率

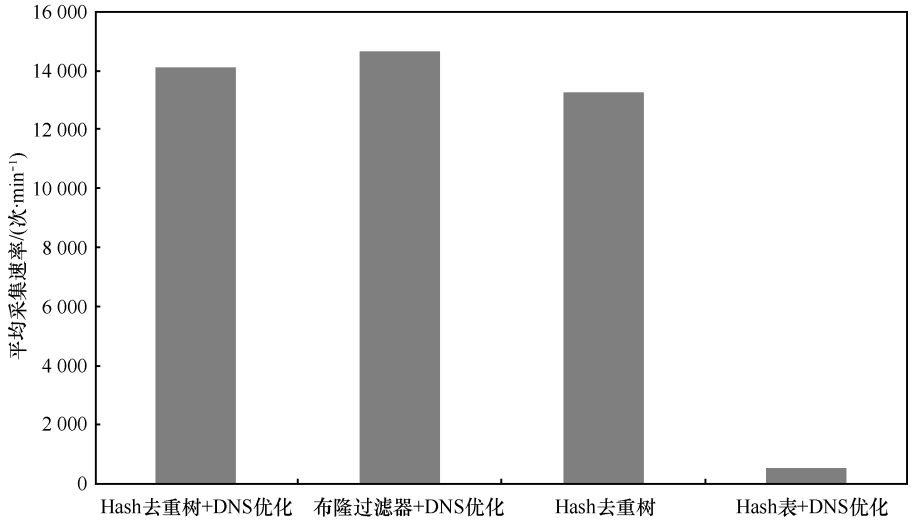


图 7　各优化项控制变量下对比结果

针对 IP 地址限制网站，本文设计了一种基于域名的 IP 地址代理调度模型，使 IP 地址代理池可以被高效利用。在灵活性方面，爬虫设计了一套通用的解析规则，解决了多种类型页面的解析需求。

参考文献：

[1] NEGUS C. Docker containers from start to enterprise (includes content update program): build and deploy with Kubernetes, Flannel, Cockpit and Atomic[J]. Vaccine, 2016(19):S87-S95.

[2] 严华云,关佶红. Bloom Filter 研究进展[J]. 电信科学,2010, 26(2):31-36.
YAN H Y, GUAN J H. Survey of Bloom Filter[J]. Telecommunications Science, 2010, 26(2):31-36.

[3] 吴桦,龚俭,杨望. 一种基于双重 Counter Bloom Filter 的长流识别算法[J]. 软件学报,2010, 21(5):1115-1126.
WU H, GONG J, YANG W. Algorithm based on double Counter Bloom Filter for large flows identification[J]. Journal of Software, 2010, 21(5):1115-1126.

[4] 张进,邬江兴,刘勤让. 4 种计数型 Bloom Filter 的性能分析与比较[J]. 软件学报, 2010, 21(5):1098-1114.
ZHANG J, WU J X, LIU Q R. Performance evaluation and comparison of four Counting Bloom filter schemes[J]. Journal of Software, 2010, 21(5):1098-1114.

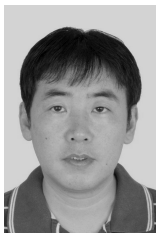
[5] 严磊,丁宾,姚志敏,等. 基于 MD5 去重树的网络爬虫的设计与优化[J]. 计算机应用与软件, 2015(2):325-329, 333.
YAN L, DING B, YAO Z M, et al. Design and optimisation of md5 duplicate elimination tree-based network crawler[J]. Computer Application and Software, 2015(2):325-329, 333.

[6] BRIN S, PAGE J. The anatomy of a large-scale hypertextual web search engine[J].Computer Networks and Isdn Systems, 1998,98 (30) : 107-117.

[7] 朱泽德,李淼,张健,等. 基于文本密度模型的 Web 正文抽取[J]. 模式识别与人工智能,2013(7):667-672.
ZHU Z D, LI M, ZHANG J, et al. Web content extraction based on text density model[J]. Pattern Recognition and Artificial Intelligence, 2013(7):667-672.

[作者简介]

石恩名（1991−），男，现就职于广州优亿信息科技有限公司，主要研究方向为数据挖掘、人工智能和地理信息系统等。



肖晓军（1970−），男，博士，广州优亿信息科技有限公司高级工程师，主要研究方向为大数据、数据挖掘和电信行业应用等。

卢宇（1983−），男，现就职于广州优亿信息科技有限公司，主要研究方向为大数据、机器学习和人工智能等。