分布式网络爬虫 URL 去重策略的改进

吴小惠
(福建交通职业技术学院,福建 福州 350007)

摘　　要:分布式网络爬虫作为一门新兴技术,已经应用在一些大型商业的搜索引擎系统当中.重点放在分布式技术在网络爬虫领域中, URL去重这一分布式网络爬虫的核心问题上,以基于内存的去重方式为基础,扩展改进传统的广义表数据结构,提出了一种新的基于内存改进广义表的 URL去重算法.这种算法与传统的去重算法相比较,在空间效率可行范围之内,有效地缩短了单次去重的时间,使总控服务器上的去重不再成为整个系统的瓶颈.

关　键　词:网络爬虫;分布式;URL去重;广义表
中图分类号: TP393　　**文献标识码**: A　　**文章编号**: 1673 - 1670(2009)05 - 0116 - 04

1 引言

分布式网络爬虫作为一门新兴技术,拥有广阔的应用前景.目前已经应用在一些大型商业的搜索引擎系统当中,但由于目前搜索引擎核心技术仅仅掌握在少数寡头公司手中,许多理论上和技术上的难点有待突破,应用上也有一些实际问题需要解决.

1.1 什么是 URL 去重

首先,我们必须理解什么是 URL去重, URL去重的作用是什么?众所周知,互联网中链接数量庞大,独立的连接有上百亿个,当爬虫爬行时,难免会遇到重复的将相同的 URL加入到等待爬行队列当中,这样不可避免的将使爬虫作重复无用的工作,大大降低了爬虫的效率.因此,对爬行的链接进行下载之前,我们要分析网页是否已经存在于爬行队列当中,以免重复,这个过程就是 URL去重. URL去重是 spider中最麻烦的一个地方了,就算是最小的爬虫, URL库也是百万数据量级的.如果去重的速度不够快,会极大的影响下载的速度[1].因此, URL去重是分布式爬虫系统的核心内容.

1.2 常用的 URL 去重的方案的不足

URL的去重操作通常有以下几种常见的方式,下面我们详细介绍各种去重方式的优劣.

1) 基于数据库的去重方式

所谓基于数据库的去重方式,就是所有的 URL链接存储于数据库当中,在对 URL进行去重的过程中,需要遍历数据库的每条记录,如果查找到与指定 URL相同的记录,则认为 URL应该去重,否则,将 URL加入到任务队列当中.同时对于爬行结束的 URL,应该依次的插入到数据库当中,表示该 URL已经被爬行过[2].基于数据库的去重方式是比较常用的一种分布式网络爬虫的去重方案,因为它的 URL去重稳定性非常高,也非常适合 URL的任务调度.但是基于数据库的去重方式却有一个非常致命的缺陷,就是去重效率非常低,当数据库的记录数超过百万时,数据库的性能会急剧下降.所以,基于数据库的去重方式的时间效率成为了整个分布式系统的瓶颈.

2) 基于内存的去重方式

基于内存的去重方式,只用内存去重,每次开始下载时把 URL写到内存的 hash表中,然后通过 hash表去重,这方案看上去很快,但有个严重问题,内存不可能这么大,内存资源很快就会被耗光,导致整个分布式系统的任务调度出现瘫痪.

3) 基于磁盘路径的去重方式

基于磁盘路径去重的核心思想是,对于爬行得到的链接,首先对其进行 MD5编码,得到 32位长的 MD5值,然后将 MD5值分割为 32位的数组,根据分割的顺序,依次在磁盘当中创建相应的目录,

收稿日期: 2009 - 06 - 10
作者简介:吴小惠(1971 -),女,福建省沙县人,福建交通职业技术学院信息技术与工程系副教授,硕士,主要研究方向:数据库管理.

这样一个 URL 将创建一个深度为 32的磁盘路径. 当新链接到来时,首先检查其 MD5值对应的路径是否存在,如果存在则进行去重操作,否则创建其 MD5值对应的路径[3]. 此种基于路径的去重方式具有较高的时间效率,但是不适用于一个面向全网爬行的大规模分布式爬虫系统,因为随着爬行的深入进行,URL 数量不断增大,对应每个 URL 都创建了相应的路径,将使系统碎片增大,最终会导致总控服务器的磁盘系统崩溃瘫痪的后果. 但是对于小规模针对局域网的网络爬虫来说,这种基于磁盘路径的去重方式,不失为一个效率比较高的去重方式.

4) 基于布隆过滤器的去重方式

在网络爬虫里,判断一个网址是否被访问过. 最直接的方法就是将集合中全部的元素存在计算机中,遇到一个新元素时,将它和集合中的元素直接比较即可. 一般来讲,计算机中的集合是用哈希表 (hash table)来存储的. 它的好处是快速准确,缺点是费存储空间.

使用布隆过滤器,它只需要哈希表 1/8到 1/4的大小就能解决同样的问题. 布隆过滤器是由巴顿·布隆于 1970年提出的. 思路是:用一个 16倍大的地址空间,让所有 8个 hash函数都映射到这个地址空间里面. 布隆过滤器的好处在于快速、省空间. 但是有一定的误识别率. 常见的补救办法是再建立一个小的白名单,存储那些可能被误判的邮件地址[4].

2 基于内存改进广义表数据结构的去重算法研究

综合以上多种 URL 去重的方法,结合分布式网络爬虫的实际需要,笔者对以上几种方法加以结合和改进,提出了一种基于内存存储的改进的广义表去重的算法,这种算法大大减少了 URL 去重的单次时间,一定程度上提高了整个系统的 URL 去重效率[5].

2.1 改进的广义表存储结构

广义表,被广泛的应用于人工智能等领域的表处理语言 LISP语言中. 作为 LISP语言中一种最基本的数据结构,广义表的定义是递归定义的[6].

我们发现,传统广义表定义了 2种类型的节点,即普通节点和元素节点,这表明,一个广义表一旦创建,则不能进行动态的添加,除非我们先将叶子结点修改为普通节点,这样便浪费了大量的时间,因此我们有必要对传统广义表数据结构进行修改,使广义表能够动态扩展,适应去重的需要[6].

笔者改进的广义表存储结构的存储形式是将 URL 字符串分割为单个的字母存储,广义表数据结构当中的 data即为单个的字母,例如 sina. com 这个 URL,将其分割成 s, i, n, a,. , c, o, m这 8个单个的字母存储成一个广义表.

定义如下的数据结构,实现以上的需求:

```
/改进广义表的头尾链表存储结构
public class GLTree {
private GLNode root;
public int totalpoint;
public GLTree() {
root = new GLNode();
totalpoint = 0;
}
}
/改进广义表的头尾链表节点数据结构
class GLNode{
public char data;
public GLNode head;
public GLNode tail;
public boolean isRoot;
public GLNode() {
this. data = '?';
head = tail = null;
isRoot = true;
}
public GLNode(char data) {
this. data = data;
head = tail = null;
isRoot = false; }}
```

由以上的数据结构我们可以发现,与传统的广义表相比,改进后的广义表,每个节点地位相等,不分元素节点和非元素节点,同时,对于每个结点,增加了一个判断是否为 root节点的标志. 对于 sina. com, sohu. com这 2个 URL 的网址,我们构造改进的广义表结构图如图 1:

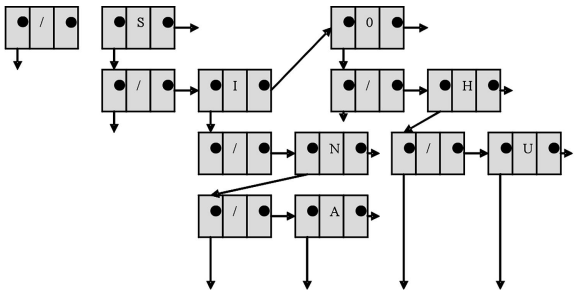

图 1 改进广义表结构示意图

2.2 基于改进广义表的 URL 去重算法

针对以上定义的改进的广义表数据结构,我们就可以根据广义表的遍历算法,针对一个新的 URL 进行遍历去重,对于遍历不存在的 URL,我们便可以将其插入到已经构造好的改进广义表当中.

1)动态插入改进广义表

当分布式网络爬虫系统的每个附属服务器分析爬行一个 URL 结束后,这个附属服务器会将该 URL 发送到总控服务器当中,总控服务器负责将此 URL 记录到内存当中的改进广义表当中,这个过程即是动态插入改进广义表的过程[6].

由于所有的 URL 网址集中在总控服务器中保存,因此整个插入过程都是在总控服务器当中进行. 首先,从改进广义表的 root 节点出发,总控服务器从第一个字母开始遍历每一个 URL 的字母,如果这个字母对应的节点存在,则 GLTree 在这一个节点执行 goHead()操作,转向 head 指针进入下一层继续执行遍历,相反,如果在遍历过程当中,某一个字母对应的广义表节点不存在,则需要先创建该节点对应层上的 root 节点和该字母的对应节点,然后将 root 的 head 指针指向该字母的节点,最后返回,继续遍历 URL 字符串,直到 URL 所有字母遍历结束. 该过程的流程图如图 2.

在实际编写程序过程当中,我们借用了一个 current 指针来完成整个 URL 的循环创建,整个过程当中,改进广义表就像一个动态增加的大树,而 current 则是忙碌于从树根到各个节点的小虫,不停的向上发展,而每一个 URL 来到后,都会有一个 current 这样的小虫负责将整个 URL 创建成为这个大树的一个树叶.

2)针对改进广义表的去重操作

URL 的去重是整个系统的核心内容,基于以上改进的广义表数据结构,使高效率的 URL 去重变成可能. 去重的过程与上面介绍的动态插入广义表过程十分类似,总控服务器从内存当中的动态广义表根节点出发,遍历待去重的 URL 每个字母,如果当前的字母对应的节点不存在,这说明该 URL 没有被爬行过,可以将其路由到等待队列当中,相反,如果字母遍历到整个 URL 的最后一个字母,依然存在对应的节点,则该 URL 已经爬行过,说明该 URL 需要去重.

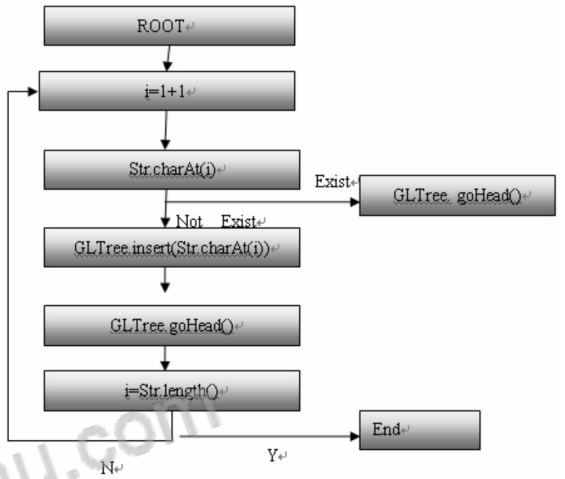

图 2 改进广义表动态插入流程图

整个去重的流程图如图 3.

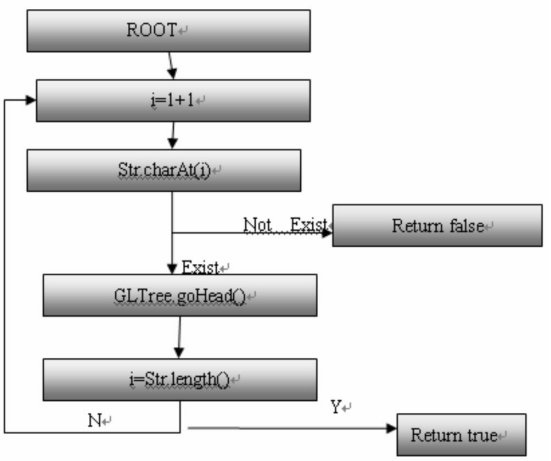

图 3 改进广义表去重流程图

3 改进广义表算法的时间和空间效率

3.1 时间效率

通过对以上数据结构的程序实现,进行数据实验,试验过程当中我们采取了 1万 6千条 URL 作为试验数据,最终得出的 URL 去重的时间表如表 1.

表 1　时间效率表

去重方式	耗时 /ms
数据库	968
内存链表	15
改进广义表	0.8

试验采用迅驰 1.86G,1G内存的计算机,并于 java环境测试,由于改进广义表的去重速度非常快,Eclipse无法计算到单次去重毫秒级别的耗时,因此我们进行了 20次去重操作,最后得到平均耗时为 0.8ms,可见,在时间效率上面,基于改进的广义表的方式时间效率大大优于基于数据库的去重方式和基于内存链表的去重算法.

3.2空间效率

但是,除了较高的时间效率,我们还应该考虑到空间效率.毕竟内存不会像硬盘一样容量巨大,因此,基于改进的广义表算法去重方式必须具有可行的空间效率才能最终被采用.

实验数据中,当有 1万条 URL数据的时候,占用的内存大小为 2MB,那么我们假设在此之后的增加为线性增加,假设一共有 1千万个网站需要爬行,则一共需要 2GB的内存.而实际存储过程当中,由于公用节点现象的普遍存在,实际占用的内存空间应该比 2GB小,因此,只要主控服务器的内存大于 2GB,整个系统的正常运行是可能的.可见,采用改进广义表算法作为去重算法,具有较高的空间可行性.

4结论

综上,在 URL去重这一分布式网络爬虫的核心问题上,系统实现上述功能模块,采用 Java编写,以基于内存的去重方式为基础,在通过对网页内容及爬行过程当中 URL队列的保存采用内存改进广义表的形式,扩展改进传统的广义表数据结构,设计新的基于内存改进广义表的 URL去重算法.在一定程度上避免了采用通用大型数据库 (如 Oracle、DB2)的时空开销.同时,在算法设计上一定程度缓解了总控服务器上的去重对于整个系统的瓶颈问题,提高一些时间和空间效率.

参考文献:

[1]周立柱,林 玲. 聚焦爬虫技术研究综述 [J]. 计算机应用,2005,23(9).

[2]张　军. 分布式系统技术内幕 [M]. 北京:首都经济贸易大学出版社, 2006.

[3]池　静. Bloom Filter和 Weighted Bloom Filter的比较与研究 [J]. 河北师范大学学报:自然科学版, 2002,16(2).

[4]肖明忠,代亚非,李晓明. 拆分型 Bloom Filter Split Bloom Filter[J]. 电子学报,2004(2).

[5]田春峰. Url排重 Bloom Filter算法、误差及其他 [EB/OL] (2007-01-23). http://blog.csdn.net/accesine960/archive/2007/01/23/1491483.aspx

[6]严蔚敏. 数据结构 [M]. 北京:清华大学出版社,2002:113-120.

Improvement on Unrepeated Tactics of URL of Distributed Spider

WU Xiao-hui

(Fujian Communications Technology College, Fuzhou, Fujian 350007, China)

Abstract: As a new technology, distributed web spider has been widely applied to some great commercial search engine systems. The stress is laid on the core problem of distributed web spider - Unrepeated URL. Based on the memory mode Unrepeated URL, the traditional generalized lists data framework is expanded and improved. A new Unrepeated URL algorithm based on the memory improved generalized list is put forth. Compare with the traditional Unrepeated algorithm, this algorithm can effectively improve the time of single detecting near-duplicate under the approval range of space efficiency, which makes the Unrepeated URL in the general control server impossible to become the bottle-neck of the whole system.

Key words: wob spider; distributed; unrepeated URL; generalized lists

---

阅读此文的还阅读了：

1. 基于Bloom Filter的网络爬虫URL消重算法研究
2. 论网络爬虫搜索策略
3. 两级管理模式下国土资源信息系统运行模式
4. 一个网络环境下多媒体CAI软件平台
5. 基于网络分布式称重的烟丝振动分选系统的研究
6. Ponder策略语言基于事件描述的扩展
7. Ponder策略语言基于事件描述的扩展
8. 基于GNP算法的分布式爬虫调度策略
9. 一种改进的T-Spider分布式爬虫
10. 面向网络的数据挖掘技术浅析
11. 一种基于局域网的事务协同与分布处理方法
12. Delphi网络应用程序性能分析
13. 基于网络环境的地理信息系统整合与知识发现
14. 物理分散 网络联动 分布式配置——“十二五”期间我国人防工程建设将实现重大转变
15. 分布式医疗保险管理系统的分析与设计
16. 网上考试策略研究和分布式系统的实现

17. 基于网络数据库的列车运行图体系结构研究

18. 分布式光伏发电微网控制策略探究

19. 网络安全分布式防火墙

20. Nutch分布式网络爬虫研究与优化

21. IBM—PC与MCS—51单片机分布式控制网络研究

22. 高校网上考试策略研究和分布式系统的实现

23. 分布式系统的网络安全性分析及策略

24. 基于信任管理系统的分布式访问控制机制

25. 分布式网络爬虫URL去重策略的改进

26. 分布式虚拟环境DVENET研究进展

27. BITBUS为基础的分布式监控系统

28. 禁止无用文件随意解压

29. 一种分布式总线型高速数据采集网络系统

30. 基于P2P和策略的分布式网络管理模型

31. LONWORKS控制网络技术及其应用

32. 棉纺织厂织机数据采集监测系统的升级

33. 高校分布式网上考试系统的分析与研究

34. 无线传感器网络中一种分布式声源定位方法

35. 分布式网络爬虫的设计与实现

36. 分布式系统中进程迁移的一种实现方法

37. POWERLINK助阿尔斯通电力实现网络技术标准化

38. 分布式网络爬虫系统的任务调度策略改进

39. ControlLogix系列控制器在连铸自动化系统中的应用

40. 大型分布式管理信息系统的安全问题研究

41. 基于网络数据库的列车运行图体系结构研究

42. 分布式陆军物资供应信息系统的研究与展望

43. 分布式企业面临的网络挑战

44. 模拟现场监控教学实验室分布式系统的设计

45. 基于TCP/IP的分布式网络视频监控系统

46. 网络爬虫在网页信息提取中的应用研究

47. 泸州电视台播出系统数字化改造的探讨

48. 基于蓝牙技术的智能家居网络技术

49. 网络系统的分布式共享内存

50. 基于软件芯片技术的开放式数控故障诊断系统