分布式 Web Crawler 的研究:结构、算法和策略

叶允明,于 水,马范援,宋 晖,张 岭
(上海交通大学计算机科学与工程系,上海 200030)

摘 要: 本文介绍了一个大型分布式 Web Crawler 系统——Igloo 1.2 版.它采用分布式的系统结构,通过我们设计的二级哈希映射算法使系统可以进行高效的任务分割,并且系统的规模动态可扩展.爬行网页的质量是评价 Crawler 的一个重要指标,Igloo 以 PageRank 值作为网页质量评价的标准,从而提高了爬行质量.加快爬行速度的关键是如何解除 Crawler 系统中的性能瓶颈,本文对此也作了详细的讨论,并提出了一种基于"滞后合并"策略的 URL 数据库存取方法.实验表明,Igloo 在保持高性能的同时能快速爬行到高质量的网页.

关键词: Web 爬虫; 爬行策略; 分布式系统
中图分类号: TP391; TP393 **文献标识码:** A **文章编号:** 0372-2112 (2002) 12A-2008-04

On Distributed Web Crawler: Architecture, Algorithms and Strategy

YE Yun-ming, YU Shui, MA Fan-yuan, SONG Hui, ZHANG Ling
(*Department of Computer Science and Engineering, Shanghai Jiaotong University, Shanghai* 200030, *China*)

Abstract: We describe a large - scale distributed Web Crawler system, i.e. Igloo V1.2. Igloo's distributed architecture is based on our two-tiered Hash mapping algorithm, so that it can do efficient task partition while at the same time providing dynamic scalability. As the quality of crawled Web pages is an important factor for evaluating crawlers, it employs PageRank value as the evaluation metric of pages to improve its crawling efficiency. This paper also provides a detailed discussion of the performance bottlenecks in crawler systems, and proposes a new URL repository access method based on "delayed merging" strategy to enable high-speed crawling. The experiments show Igloo can quickly crawl high-quality Web pages as well as present high performance.

Key words: Web crawler; crawling strategy; distributed system

1 引言

随着互联网的爆炸性增长,Web 已经发展成为站点遍布全球的巨大信息服务网络.据 SearchEngineWatch[4] 预计,目前 Web 上的网页数目已超过 20 亿个,并且以每天新增一百万个页面的速度增长.面对如此巨大的信息库,如何快速准确的检索到自己需要的信息呢? 搜索引擎已经成为 Web 信息获取的一种最重要的手段.Web 爬虫 (Crawler)是搜索引擎的信息搜集代理,沿着超链遍历 Web,搜集信息资源.

索引网页数量的大小、质量是评价一个搜索引擎好坏的重要指标.因此,一个高效的 Crawler 系统是一个好的搜索引擎的重要基础.出于商业机密的考虑,目前各个搜索引擎使用的 Crawler 系统的技术内幕一般都不公开.现有的文献也仅限于概要性介绍,如斯坦福大学的 Google[1,7],康柏研究中心的 Mercator[2],卡内基梅隆大学的 WebSPHINX[5],对于分布式 Web Crawler 的详细设计和实现很少涉及.Mercator 和 WebSPHINX 是在单机上运行的,它们的信息搜集速度比较有限,就目前互联网的规模来说,已无法在一个有效的时间范围内完成一次搜集整个 Web 的任务.分布式 Crawler 系统采用多机并行工作,提高整个系统的工作效率,并具有良好的可扩展性,是必然的发展趋势.

本文以我们自主设计开发的纯 Java 分布式 Web Crawler 系统 Igloo 1.2 版为基础,详细介绍它采用的一些关键技术和算法.

2 分布式 Crawler 系统结构

Igloo 采用分布式的系统结构,它由分布在不同的计算机上的多个 Crawler 组成,各个 Crawler 并行协同工作.

2.1 Crawler 的结构

单个 Crawler 的爬行(Crawling)过程如图 1 所示.每个 Crawler 从一组种子 URL 开始,由 DNS 解析器获得该 URL 对应的主机 IP 地址,然后通过机器人拒绝协议检测后由 HTTP/HTTPS 下载模块下载该网页.URL 抽取器从下载的网页中抽取出新的 URL,然后由 URL 过滤器逐个检测 URL 是否符合过滤规则的限制.最后,用哈希函数计算各个 URL 的哈希值,如果属于本 Crawler 的下载范围,则将该 URL 加入到本地 URL

收稿日期:2002-06-02;修回日期:2002-09-12
基金项目:国家自然科学基金(No.60221120145);上海市科委基础研究重点项目(No.02DJ14045)

数据库中;否则把该 URL 插入到 URL 发送队列中,由 URL 分发器定时传送给对应的 Crawler.系统中,爬行策略控制器按照爬行策略将待爬行的 URL 排序,使重要的 URL 资源得以优先下载.

```
crawlingFunc(urlQueue, seedUrls){
  while(not isEmpty(urlQueue)){
    curUrl = dequeue(urlQueue);
    hostIp = dnsResolve(getHostName(curUrl));
    if(not isRobotExcluded(curUrl)){
      page = downlaodPage(formatUrl(curUrl));
      foundUrls = extractUrls(page);
      for each url in foundUrls{
       if(not isFiltered(url)){
         if(isUncrawled(url)){
           enqueue(urlQueue,url);
           reorderQueue(urlQueue);
         }
       }
       else{
         routeUrl(url)
       }}}}
}
```

图 1　爬行过程伪代码

2.2　分布式任务分割算法

如何将巨大的爬行任务均衡分配给各个 Crawler 是分布式 Web Crawler 的关键问题之一.目前许多 Crawler 系统都采用集中式的任务分割策略,即系统有一台主机作为协调器,它采用哈希算法为各个 Crawler 分配 URL 爬行范围.每个 Crawler 对新发现的 URL 都要用哈希函数进行判断,把不属于本 Crawler 爬行范围的 URL 上传到协调器,由协调器根据 URL 的哈希值转发给对应的 Crawler.这种策略的最大缺点是:协调器要转发大量 URL,将成为系统性能的瓶颈.

为解决这个问题,Igloo 通过二级哈希映射算法支持分布式 URL 分配,并使系统规模动态可扩展.二级哈希映射如图 2 所示.首先,在一次爬行周期内系统的最大规模一般是可以确定的,则假设最大 Crawler 数目是 M,而系统当前运行的 Crawler 数目是 N,显然 $N \leqslant M$.相应的每个 Crawler 有两张表:一张是逻辑节点表,存储 M 个逻辑节点的信息,每个表元素如果物理上对应的 Crawler 存在,则它的值为该 Crawler 的 ID 号,否则为零;另一张是物理节点表,每个表元素存储系统当前各个 Crawler 的信息.经过标准化的 URL 经第一次哈希运算映射到相应的逻辑表元素上,如果该元素不为零,则取出该 ID 号,判断是否需要将该 URL 路由给其它 Crawler;如果该元素为零,说明该逻辑节点目前还没有对应的物理 Crawler,则要把映射到该元素上的所有 URL 均分到系统当前的各个 Crawler 上,这是通过第二次哈希映射来完成的.经第二次哈希映射,获得对应 Crawler 的 ID 号,最后判断是否路由该 URL.可以验证,系统运行过程中 N 可以动态变化,而不会影响 URL 的均衡分配.

要想把 URL 尽可能平均的分配到每台 Crawler 机器上,需要选择好的哈希函数.为此,我们设计了一个 URL 哈希函数,它的函数表达式如下:

$$Hash(S) = \sum_{i=1}^{S.length} \mathrm{Ascii}(S.charAt(i)) \bmod P$$

其中,在第一次哈希映射中 $P = M$,在第二次哈希映射中 $P = N$;Ascii 函数取字符串中每个字符的 ASCII 码.经过我们大量实验,证明该函数可以产生相当平均的哈希分布,保证各台机器上的 URL 队列长度均匀增长.

3　基于 PageRank 的爬行策略

由于 Web 信息的海量性,即使几个最大的商业性搜索引擎,如 Google,Yahoo! 等,它们各自的索引网页数也不超过整个 Web 网页数的 40%[4],因此在有限的时空资源下,Crawler 要采用合理的爬行策略,优先搜集质量高的网页.

3.1　网页质量评价标准

整个 Web 逻辑上可以看作是一张有向图 $G = (V, E)$,V 表示节点集合,对应于各个网页,例如网页 P、Q 对应于节点 V_i 和 V_j,$V_i, V_j \in V$;E 表示有向边集合,对应于网页之间的超链接,即如果网页 P 有一个超链接指向网页 Q,则 P 与 Q 之间就有一条有向边 (V_i, V_j).因此 Crawler 的爬行过程可以看成是有向图的遍历过程,以下讨论的爬行策略和算法将基于这个图结构.根据超链接结构的特点,一个网页 P 的质量 $I(P)$ 可以用两种方法度量.首先定义一个引链数的概念.

定义 1　引链数:假设节点 V_j 对应于网页 Q,则网页 Q 的引链数就是指向 V_j 的有向边数目,即链接到 Q 的超链接数目.

基于引链数的评价标准:按照这种标准,网页的引链数越多则网页的质量越高,类似于科技论文的引文统计.

基于 PageRank[3,8]的评价标准:基于引链数的评价没有考虑引链页面的质量差异对目的节点的影响,所有引链页面都同等对待.这对于 Web 网页来说是不合理的:假设网页 Q1、Q2 分别有一个引链 P1→Q1,P2→Q2,且 I(P1) > I(P2),则显然 $I(Q1) > I(Q2)$,而在基于引链数的评价中 $I(Q1) = I(Q2)$.也就是说引链页面质量越高,则相同引链数条件下被链接网页的质量也越高.PageRank 算法就是基于这个假设的.

PageRank 算法原先被 Google 搜索引擎用于查询结果集的相关度排名.Google 给每个索引网页赋予一个 PageRank 值,值越高的网页认为它的质量越高.它的计算公式如下:

$$R(p) = \varepsilon/n + (1-\varepsilon) \cdot \sum_{(q,p)\in G} R(q)/outlink(q)$$

其中 G 是表示已搜集网页集的有向图;p、q 是 G 中的节点,q 是 p 的引链页面;n 是 G 中的节点个数,即网页总数;ε 是衰减因子用来解决局部排名下陷问题[8].PageRank 值是通过多次迭代计算获得的,使 PageRank 值满足收敛性.

3.2　分布式 PageRank 值计算

Igloo 1.2 系统以 PageRank 值作为评价待爬行网页质量的标准,每个待爬行 URL 都有一个 PageRank 值,由爬行策略控制器根据 PageRank 值对它们进行排序,使 PageRank 值高的

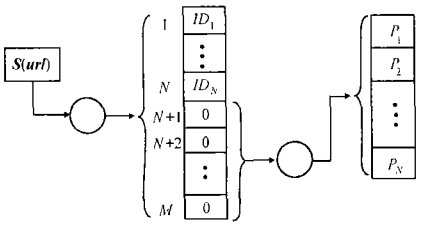

图 2 URL 二级哈希映射

URL 得以优先爬行.与 Google 搜索引擎的离线、集中式计算不同,Igloo 中的 PageRank 值是在线、分布计算的.每个 Crawler 在爬行过程中根据全局的引链信息动态计算.系统在爬行中将 URL 数据库分为两部分,一部分是已经爬行过的 URL 集合 C,另一部分是待爬行的 URL 集合 W,根据 C 中的引链信息计算出 W 中各个元素的 PageRank 值.由于采用分布式的系统结构,计算过程的关键问题就是各个 Crawler 如何收集全局的引链信息,使 Crawler 可以在本地计算属于自己爬行范围的待爬行 URL 的 PageRank 值.为此,每个 Crawler 在本地维持一张引链表,当从下载的网页中抽取出 URL 时要将属于自己爬行范围的引链信息写入引链表,对不属于自己爬行范围的 URL,在路由给其它 Crawler 时也要同时传递引链信息,这样每个 Crawler 就拥有了全局的引链信息,可以在本地进行 PageRank 计算.随着爬行过程的进行,集合 C 和 W 都将不断变化,要通过连续的迭代计算使 PageRank 值保持准确.

4 高速爬行

通常,要使已爬行的网页库保持一定的新鲜度,Crawler 的爬行周期应不超过 15 天.按照目前的 Web 规模,即使只爬行全部网页的 30%,这个时间限制对 Crawler 也是巨大的挑战.为此,Igloo 采取了一些性能优化策略,如 DNS 缓存、多线程机制等,其中 URL 数据库的存取是核心问题.

4.1 URL 存取性能瓶颈

Crawler 运行时要不断的访问 URL 数据库,然而 Crawler 处理的 URL 是百万以上的数量级,URL 数据库具有海量的特点,使 URL 存取操作成为系统性能的瓶颈.URL 数据库是以大数据结构形式存在的,要实现快速存取,最好的方法是将整个数据结构都存储在内存中.但是,按照平均一个 URL 40 字节计算,5 千万个 URL 已经需要 2G 字节的内存空间,且随着系统规模的扩展,需要的内存空间将迅速增加.因此,采用磁盘文件和内存交换的方式来存储大数据结构是更好的选择.由于磁盘存取速度与内存存取速度差几个数量级,因此 URL 的存取就成为系统性能的瓶颈,这就需要考虑采用高效的存储交换策略,保证 URL 存取的高性能.

4.2 URL 大数据结构与存储交换

对应于每个 URL,URL 数据库实际上包含了两部分内容,一部分是 URL 字符串本身;另一部分是 URL 的元数据,即 URL 的访问标志位和 HTTP1.1 协议中规定的 URL 资源的最近修改日期.URL 数据库是以 Trie[6] 数据结构实现的.在 URL Trie 中,URL 字符串本身存储在从根节点到叶子节点之间的内部节点中,而 URL 的元数据存储在叶子节点上.由于 Trie 对关键字的检索速度只与检索字符串的长度有关,通过该数据结构可以实现快速的 URL 检索.经测试,其检索速度约为 4000 个 URL/秒.

由于 URL 数据结构是采用磁盘文件的形式存储,通过内存-磁盘数据结构存储交换实现 URL 的存取,因此提高存取速度的关键是如何尽量避免磁盘存取,使大部分 URL 存取操作可以立刻在内存中完成.Igloo 采用的设计原则是:尽量保证从 URL 存取客户的角度看,URL 存取操作是立刻在内存中完成的,使 URL 存取客户不会因磁盘 I/O 而阻塞.相应的,Igloo 的 URL 存取策略是采用"滞后合并"的方法:开始时将抽取出的 URL 以 Trie 数据结构的形式存储在内存中,当该数据结构占用的内存空间超过一定的大小时,才将该内存驻留的 Trie 与磁盘文件中存储的 Trie 数据结构进行合并,在合并时进行 URL 的检索比较和插入.可以看出,原来的 URL 重复性判断和插入操作都是在内存驻留的"局部"Trie 中进行的,最终插入到"全局"Trie 的操作被滞后了.经过反复实验表明,这种方法具有高效的存取性能.

5 实验

5.1 性能测试

我们使用了 5 台安装了 RedHat Linux7.0 和 JDK 1.3 的计算机,每台计算机上运行一个 Crawler.另外,有一台安装 Windows 2000 的计算机运行系统管理平台.所有机器通过内部 100Mbps 交换机连接到一个 10Mbps 的 CERNET 网关.

我们选择了近 20 个国内大学的主页作为种子 URL,在教育网内部作了测试.考虑到校园网络负载的限制,我们限定每个 Crawler 最多同时启动 5 个线程.运行约 11 个小时后,得到表 1 的测试数据.在 CERNET 的测试中,下载失败的 URL 数目约占下载总数的 17%,远远高于文献[2]中统计的结果.我们认为这主要是因为 CERNET 里的 Web 服务器管理较为松散,相当一部分网站长期无人维护造成的.

表 1 Web Crawler 性能统计数据

URL 数据库大小	2,984,503 个 URL
下载 URL 总数	1,855,172
下载失败的 URL 总数	316,973
平均下载速度	46.85 URL/s;467.21KB/s

在性能方面,文献[1]显示 Google 的 Crawler 系统使用 3 台机器,每台机器同时建立 300 个 HTTP 连接,它的最快下载速度约为 100 URL/s.由于在测试中我们 5 台机器的并发连接总数为 25,而且我们的网络带宽大大小于 Google 系统,由此可见,Igloo 在性能上已达到了较高的水平.

5.2 爬行网页质量测试

我们也对基于 PageRank 的爬行策略进行了实验.该实验以上海交通大学校园网内的各个 Web 服务器作为爬行目标,选取了 500 个我们认为是最重要的网页,前 5 个如表 2 所示.

可以看出,基于 PageRank 的爬行总体上比随机爬行更快找到质量高的网页.刚开始时这个差别不是太明显,这是由于这时爬行的网页数量比较少,因此 PageRank 计算的结果不够

准确,随着爬行的继续进行,基于 PageRank 的爬行很快显示出它的优越性.我们还将它的统计结果与我们的 Luka 搜索引擎计算出的 PageRank 值排序进行了比较,对于重要网页两者的排序能较好的匹配.这说明基于 PageRank 的爬行策略对爬行网页质量的提高有很大促进作用.

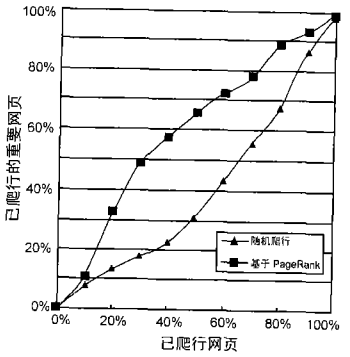

图 3　爬行网页质量实验结果

实验以上海交通大学主页为种子 URL,总共爬行了 13189 个网页,在爬行过程中不断统计爬行的重要网页个数.我们将实验结果与随机爬行策略(不对 URL 队列进行排序,即 FIFO 方式)的结果进行比较,实验结果如图 3 所示.

表 2　重要网页示例

序号	URL	网页描述
1	http://www.sjtu.edu.cn	上海交大主页
2	http://www.lib.sjtu.edu.cn	图书馆主页
3	http://www.gschool.sjtu.edu.cn	研究生院
4	http://jd.sjtu.edu.cn	交大焦点网
5	http://www.seit.sjtu.edu.cn/	电子信息学院

6　结论

Web Crawler 是搜索引擎、Web 信息检索系统和 Web 挖掘的重要基础.本文介绍了由我们自主设计开发的分布式 Web Crawler 系统 Igloo 1.2 版.它通过二级哈希映射算法使系统可以进行高效的任务分割,并采用了基于超链接分析算法 PageRank 的爬行策略,从而提高了爬行质量.我们还提出了一种基于滞后合并策略的 URL 数据库存取方法,实现了 Crawler 的快速爬行.实验表明,Igloo 在保持高性能的同时能快速爬行到高质量的网页.

参考文献:

[1] A Arasu, J Cho, H G Molina. Searching the Web[J]. ACM Transactions on Internet Technology, 2001, 1(1): 2 - 43.

[2] A Heydon, M Najork. Mercator: A scalable, extensible Web crawler [J]. World Wide Web, 1999, 2(4): 219 - 229.

[3] Monika R Henzinger. Hyperlink analysis for the Web[J]. IEEE Internet Compnting, 2001, 5(1): 45 - 50.

[4] http://www.searchenginewatch.com/ [EB/OL]. 2001.

[5] R C Miller, K Bharat. SPHINX: A framework for creating personal, site-specific Web crawlers[A]. In the 7th WWW Conference[C]. 1998.

[6] E Fredkin. Trie memory[J]. Comm. ACM, 1960, 3: 490 - 500.

[7] S Brin, L Page. The anatomy of a large-scale hypertexual Web search engine[A]. In the 7th World Wide Web Conference [C]. 1998.

[8] L Page, S Brin, R Motwani, T Winograd. The PageRank citation ranking: Bringing order to the Web[R]. USA: Stanford University, 1998.

作者简介:

叶允明　男,1976 年生于福建福清,上海交通大学计算机系博士生,主要研究方向为智能信息检索与 Web 挖掘.

于　水　男,1976 年生于江苏盐城,上海交通大学计算机系博士生,主要研究方向为机器学习与 Web 挖掘.

马范援　男,1942 年生于上海,上海交通大学计算机科学与工程系教授,博士生导师,主要研究方向为 Internet 信息获取,Web 挖掘,网格计算.