两种适用于中文信息搜集的 URL 散列函数的研究

Two Effective Functions on Hashing URL in Collecting Chinese Information

李村合 何淑庆 张培颖 (中国石油大学(华东)计算机与通信工程学院 东营 257061)

摘要:为了适应 Internet 信息量的快速增长,搜索引擎采用分布式技术搜集信息。采用分布式搜集技术的具体应用中主要存在两个关键问题:URL 匹配和系统负载平衡。针对现有的几种分布式信息搜集系统的设计中存在的一些不足,提出了利用 URL 分级散列进行定位和匹配的方法。根据中文网络信息的特点,分析了几个对字符串散列较好的函数,设计了两种 URL 散列函数,应用于分布式中文信息搜集系统中。实验表明,系统在减少 URL 匹配的资源消耗和提高系统负载的均衡性方面有很好的效果。

关键词:散列函数 分布式 搜索引擎 匹配 负载平衡

1 引言

随着 Internet 信息量的爆炸式增长,搜索引擎越来越受到人们的关注。然而信息量的增加也使得传统的搜索引擎中的信息搜集工作变得愈加繁重,这种矛盾促使了信息搜集思路的转变和技术的突破。如各种面向特定主题信息的搜索、个性化的信息搜索等,以缩小信息量求得良好的性能。但面向整个 Internet 信息的搜索引擎依然是当今搜索引擎的主流,即使是网络硬件的快速发展,单机集中式信息搜集已成为搜索引擎性能提升的瓶颈,分布式信息搜集技术成为搜索引擎信息搜集的首选。

采用分布式搜集技术是搜索引擎信息搜集模块的发展趋势,同时也带来了新的问题,例如:保证节点间不重复搜集网页、节点分配任务的均衡以及节点的损坏对系统的影响等。当前研究较多的有:URL 的划分策略、节点负载均衡和系统的动态可配置性。本文首先在这三个方面进行了深入地分析,并在此基础上,针对中文网页信息量的特点,提出了两种 URL 散列函数,在系统中采用分级散列的方法实现。实验结果表明,给出的 URL 散列函数能很好地对 URL 字符串进行散列,URL 分级散列的方法减少 URL 匹配对系统资源的消耗、提高了系统负载的平衡性和系统的健壮性。

2 分布式技术出现的问题

2.1 URL 匹配问题

重复链接是指同一个 URL 在网络中多次被引用的链接。当搜索引擎搜集网页时,由于网络中存在着大量的重复链接,使得对同一个 URL 搜集多次,从而造成系统资源和存储空间的浪费。传统的信息搜集系统大多使用集中式搜索,主要是通过两个 URL 数据表(已经访问过的表 visited_urls 和未访问过的表 unvisited_urls)解决重复搜集信息的问题,具体解决方法请参考文献[1]。

散列(hashing)是信息存储和查询所用的一项基本技术,它能以 O(1)时间复杂度对数据进行处理,可以很好地用于 URL 的匹配。天网的开发者通过实验评测表明:对字符串散列效果很好的 ELFhash 函数对 URL 散列效果并不是很理想[2],并给出了两种效果较好的散列函数。也有些研究中使用 MD5 算法对 URL 进行计算、处理并匹配,通过 MD5 算法为每一个 URL 求得惟一值。然而对于 MD5 算法产生的 128 位整数是不能直接使用的,需要除以存储空间的长度 N,其匹配效果并不会比一般的散列函数效果好,而且 MD5 算法会消耗大量的系统资源,造成系统整体性能的下降。

采用分布式技术的信息搜集系统和采用集中式技术的搜集系统解决网页重复链接的原理一样,只不过在分布式信息搜集系统中,每个节点单独搜集信息。但系统运行过程中,URL 匹配需要考虑到整个系统搜集到的 URL。因此可以通过数据共享和增加一个控制系统来解决这种问题。一些研究中通过在每个搜集节

点上，部署相同的 visited_urls 数据表对 URL 进行匹配，这浪费了搜集节点的存储空间和降低了 URL 匹配的成功率。

2.2 系统负载平衡问题

分布式信息搜集系统的另一个关键问题是系统负载平衡问题。理想状态是对 N 个任务，平均分布到 n 个节点上，每个节点的任务量为 N/n；另一种极端的状态是对 N 个任务，全部分布到一个节点上，其他 n－1 个节点任务量为 0。因此负载平衡问题对于分布式系统的性能至关重要，在不额外耗费单个节点系统资源的前提下，系统负载平衡性越好，分布式系统的性能越好。

在解决分布式信息搜集系统负载平衡问题上，一些研究采用的方法是将站点散列到系统的各个节点上，每个站点一旦散列到某个节点，则该节点负责这个站点的所有任务。对于存在 S 个站点，分布式系统中存在 n 个节点，一般的散列函数都能很好地将 S 个站点均匀散列到每个节点上。但网络中每个站点包含的信息量相差是非常大的，假定几个信息量大的站点散列到某个节点上，而另外的节点上的站点包含的信息量小，这样就会造成负载任务的不均衡问题。一般情况下，S > > n，所以设计中忽略该问题，但基于这种方法设计的系统在搜集后期会发生某些节点的搜集任务集中在几个包含信息量大的站点上的问题，造成系统搜集的效率急剧下降。当系统中的节点 n 变大，而针对搜集的站点 S 变小的情况下，站点包含信息量不均问题不能忽略。另外一些研究针对这种不平衡，通过对任务量进行二次散列或动态分配的方法来提高系统的负载平衡性，这种方法的缺点是需要额外的开销来进行二次散列或动态分配。

3 两种 URL 散列函数

据 2004 年中国互联网络信息资源数量调查报告，截止到 2004 年底，CN 注册下的域名总数为 432077 个，全国域名总数为 1852300 个。全国网站总数约为 668900 个[3]。全国网页总数约为 650682300 个。当前中文网址在百万数量级，中文网页在 10 亿数量级，站点的平均页面数量在 1000 以上。

针对中文网页信息，笔者给出了两种简单的散列函数，应用于分布式中文信息搜集系统中。

3.1 站点散列函数

虽然运用散列技术进行匹配的成功率无法准确预见，但大量的实验表明：设装填因子为 α，给定散列表空间 N，对 S 个站点进行散列，则 α = S/N，当 α < 0.75 的时候，散列的效果很好。基于此特性，设定站点散列函数的表空间为 2 的 21 次幂（2097152）。由于字符的 ASCII 码值的首位为扩展位，函数中取字符的 ASCII 值的后七位进行运算。

算法描述：① 选定一个站点的 URL 字符串；② 对 URL 字符串中的每位字符的 ASCII 码值进行计算（详细步骤见程序 1－3）；③ 对 URL 字符串计算后的值修正，是值介于 0 ~ 2097151 之间（详细步骤见程序 4－8）；④ 返回修正后结果值（程序 9）。

Java 语言实现如下：

```
int result(String url){
int r=0; int s=0;// 给定两个 32 位的整型变量 r,s=0(1)
for(int i=0;i<url.length();i++)// 依次获取 URL 中的字符(2)
    r=(r<<7)+(int)url.chatAt(i);// 字符的 ASCII 码值与变量 r 相加后值左移七位(3)
    s=r;// 将结果值赋值给变量 s(4)
    while(!(r==s)){// 相加后结果值与上次值的 1~21 位相等时停止(5)
        r=s&0x001FFFFF;// 获取值的 1~21 位(6)
        s=(s&0xFFE00000)>>21;// 获取值的 32~22 位,并将获取值右移 21 位(7)
        s=s+r;}// 获取原来值的 32~22 位与 1~21 位相加的结果值(8)
    return s;// 返回结果值 s(9)
}
```

3.2 路径散列函数

站点散列函数的计算值介于 0 和 2097151 之间。URL 地址格式为：scheme://host:port/path，有些研究中根据 host 定位节点，本文根据 path 定位节点，匹配过程中采取二次匹配，先进行 host 匹配，然后进行 path 匹配。考虑到每个站点的平均网页数量为 1000 多，设定 path 散列的表空间 N 为 2 的 15 次幂（32768），这个数值可以满足大多数的站点，对几个少

数的大站点超过 0.75N，再次进行动态散列。路径散列函数的实现相对站点散列函数简单，基于 path 的长度较长而对散列的表空间要求不高，函数中取字符的 ASCII 码值的后五位进行计算。

算法描述：① 选定一个 URL 的路径字符串；② 依次获取每个字符串的 ASCII 码值进行计算（详细步骤见程序 10－14）；③ 返回结果值（程序 15）。

Java 语言实现如下：

```
int result(String path){
int r=0; int s=0;// 给定两个整型变量 r,s=0 (10)
for(int i=0;i<path.length();i++){// 依次获取 URL 字符串的字符(11)
s=(s<<5)+(int)url.chatAt(i);// 字符的 ASCII 码值与 s 相加后值左移 5 位。(12)
if(i%3==2){ // 每三位字符的 ASCII 码值为一组(13)
r^=s; s=0;}// 将 r 异或 s 的值赋值给 r,s 清零(14)
}
return r^s;// 返回结果值 r^s(15)
}
```

3.3 散列函数在系统中的应用

以上设计了的两个散列函数，通过它们来解决分布式信息搜集系统中 URL 匹配和系统负载平衡的问题。在具体的应用中，改进了系统的设计方案。

分布式信息搜集系统的结构由控制节点和若干搜集节点构成。搜集节点负责本节点任务的搜集工作，控制节点控制整个搜集过程和管理整个搜集信息。系统中 URL 处理的流程：一个节点获取的 URL，通过路径散列函数定位节点，属于本节点处理的 URL，通过站点散列函数和路径散列函数依次进行 URL 匹配，如匹配成功，则舍弃此 URL，如匹配不成功，则将此 URL 添加到任务表中并发送控制节点，在控制节点处理；属于其它节点处理的 URL，由控制节点统一进行调度。控制节点上的工作为识别重复链接、识别新的站点以及构建整个系统的信息结构表。

4 系统性能分析

4.1 系统结构分析

有些研究表明：搜集过程中 URL 的解析速度大大超过 URL 任务的执行速度。因此，如何减轻搜集节点的资源开销是提高分布式系统性能的关键。本文给出的分布式信息搜集系统中的 URL 匹配集中在控制节点上，节省了搜集节点的资源开销和存储空间。对于 n 个节点的分布式系统，处理 S 个站点，存在的 URL 数量为 $m*S$，有 $m*S>>S>>n$。基于 URL 定位的方法使系统负载平衡性优于基于站点（host）散列的方法。

站点散列函数和路径散列函数在系统中的应用，使得系统具有启发式信息搜集的功能。在控制节点上，当 URL 经过站点散列后匹配不成功，则定义其所属的站点为新站点，控制节点启动对新站点的搜集工作，这种基于新站点的信息搜集在系统首次搜集时效果明显。

4.2 实验结果分析

实验主要测试 URL 匹配成功率和各节点的负载平衡，系统设置 7 个搜集节点和一个控制节点，系统爬行共获取 URL 总数为 246951，各节点的 URL 任务量负载如下表：

表 1 系统 URL 统计表

	系统	控制节点	节点 1	节点 2	节点 3	节点 4	节点 5	节点 6	节点 7
URL 统计	246951	246951	35580	35059	35368	35206	34972	35283	35483
匹配统计	2066713	1806747	38076	37408	37673	37077	35946	36745	37041

由表 1 得出系统将 URL 很好的散列到每个节点；节点上的匹配性能很好；系统的 URL 匹配主要集中在 URL 控制节点。经实验发现，系统运行中大部分时间节点对于 URL 的发现能力远远超过任务的执行能力。因此，URL 在控制节点上进行匹配后再发送到定位节点造成的延迟基本上不会降低系统的性能。

（下转第 48 页）

(上接第 44 页)

5 结论

面对日益增多的网络信息,搜索引擎所采用的分布式技术是当前研究的热点。其中围绕 URL 匹配、系统负载均衡及系统的扩展性等问题的研究较多。本文给出了两种简单的散列函数,对 URL 中的站点(host)和路径(path)分别散列求值,实现了 URL 的定位和匹配,减轻了节点的资源消耗。该方法的优点是散列函数简单易行,对 URL 的散列效果较好;缺点是需要分别对 URL 中的站点和路径进行匹配。

参考文献

1 李晓明、闫宏飞、王继民、搜索引擎 - 原理、技术与系统[M],科学出版社,2005:19 -54。

2 李晓明、凤旺森,两种对 URL 的散列效果很好的函数[J],软件学报,2004,15(2)。

3 2004 年中国互联网络信息资源数量调查报告[R],2005 年 1 月:73 -75。

4 燕彩蓉、彭勤科、沈钧毅、武红江,基于两阶段散列的 Web 集群服务器内容分配研究[C],西安交通大学学报,2005 年 8 月。

5 李学勇、欧阳柳波、李国徽、钟敏娟,网络蜘蛛搜索策略比较研究[J],计算机工程与应用,2004,40(4)。

6 李学勇、田立军、谭义红、欧阳柳波、李国徽,一种基于非贪婪策略的网络蜘蛛搜索算法[J],计算技术与自动化,2004,23(2)。