节点对等 Web Spider 设计与实现

张林才[1],张 燕[1],王红霞[2]
(1.辽宁石油化工大学 计算机与通信工程学院,辽宁 抚顺 113001;
2.北京青年政治学院 计算机系,北京 100102)

摘 要:由于互联网具有海量信息并且快速增长,提高搜索引擎的信息采集器 Web Spider 的数据采集和更新速度有重要意义。受计算资源限制,单机多线程 Web Spider 的采集速率不高。带中心节点的分布式并行 Web Spider 又容易产生中心节点瓶颈问题。利用 ProActive 网格网络并行分布计算中间件提供的主动对象技术、网络并行计算技术、自动部署机制等设计和实现了一个名为 P-Spider2.0 的节点对等的分布式并行 Web Spider,并设计了一个基于 Raibin 算法的 URL 去重算法。实验表明该 Web Spider 方便管理和部署,并且比单机多线程 Web Spider 具有更高的采集速率。

关键词:网络爬虫;ProActive;并行;分布式;节点对等
中图分类号:TP393 **文献标识码**:A **文章编号**:1673-629X(2010)03-0195-04

Design and Realization of Peer-to-Peer Web Spider

ZHANG Lin-cai[1],ZHANG Yan[1],WANG Hong-xia[2]
(1.School of Computer and Communication Engineering, Liaoning Shihua University,Fushun 113001, China;
2.Computer Science Dept., Beijing Young and Political College, Beijing 100102, China)

Abstract: With the rapid growth of Internet's massive information, it becomes very significant to enhance the data acquisition and update speed of search engines' information collector: Web Spider. Bottleneck caused by center node is apt to occur in the distributed parallel Web Spider with center node. The peer-to-peer distributed parallel Web Spider named P-Spider2.0 is designed and realized with the active object, network parallel computing technology and automatic deployment of ProActive which is a grid and network parallel distributed computing middleware. Experiments were conducted and the experimental results show that the peer-to-peer Web Spider has higher data collection rate than multi-threaded Web Spider, and is easier to manage and deploy.

Key words: Web Spider; ProActive; parallel; distributed; peer-to-peer

0 引 言

Web Spider 是一种自动下载网页的程序,一般在搜索引擎中负责采集信息,它依靠网页之间的链接关系遍历互联网,把分散存储在互联网上的信息下载到本地,以便于搜索引擎对这些数据进行分类索引。由于互联网信息量快速增长,要求 Web Spider 要有更快的采集和更新速度。在单机上使用多线程技术能够在一定程度上提高采集速度[1],但由于单机计算资源有限,通过多线程技术提高速率也是有限的。采用多机分布式并行结构,增加处理机和网络接口数量,能比单机多线程更加显著地提升 Web Spider 的采集效率[1]。

由于结构简单,实现方便等原因,现在分布式并行的 Web Spider 大都采用一个中心节点负责管理 URL 队列、分发任务,其他计算节点负责采集、解析的结构。但是由于中心节点的负担较重,当计算节点数量较多的时候,中心节点容易形成效率的瓶颈,所以这种结构的 Web Spider 的可扩展性较差,而且计算机节点和中心节点交换数据过于频繁,加之繁重的 URL 检索工作由单机来完成,容易影响整个系统的采集效率。于是对这种结构进行改进,设计开发出了节点对等的基于 ProActive 的分布式并行 Web Spider,即 P-Spider2.0。

在分布式并行计算方面,传统的基于 MPI 的技术具有程序可移植性差和配置复杂等不足[2]。如果直接用 java 开发,在多线程和分布式 java 应用程序之间还存在很大的缝隙,而且为了在多线程应用程序上构建分布式应用程序而禁止了代码重用,例如 javaRMI 和 javaIDL。为了实现把本地对象转化成可用的远程对象,要求编程人员对库中现有代码做较大修改,这给编

收稿日期:2009-07-06;**修回日期**:2009-10-29
基金项目:辽宁省自然科学基金(20052211)
作者简介:张林才(1978-),男,辽宁开原人,助教,硕士,研究方向为网络与并行计算、搜索引擎;张 燕,教授,博士,主要从事计算机应用的研究。

程人员增加了很大的负担。ProActive 中间件是一个基于 java 的分布并行软件包,具有 java 良好的兼容性和面向对象的可重用性,用其设计开发分布式并行程序可以很好地弥补这些不足。ProActive 还提供了使用各种网络网格中间件的接口,能方便地在网络网格环境下进行部署,使得 ProActive 在开发分布式并行 Web Spider 方面更具有独特的优势。

1 ProAcitive

ProActive 是一个由法国的 INIRA 的 Denis Caromel 教授带领的开发小组开发的适合并行、分布和并发计算,在统一框架具有的移动性和安全性的 java 开源开发包,是 ObjectWeb consortium 开源中间件的一部分[3],具有如下主要特性。

1.1 主动对象

主动对象(Active Object, AO)是 ProActive 计算概念的核心[4]。它包括一个远程对象和一个线程。这个线程控制主动对象的活动,以及和其他已经部署好的主动对象协同工作。主动对象是在标准对象的基础上增加了位置透明、活动透明和同步三种功能。主动对象的通讯默认是异步模式的。一个主动对象包括一个主要的对象、一个线程、待处理请求队列。

1.2 异步调用

ProActive 对主动对象的异步调用是通过 Future 对象来实现的。Future 对象是 ProActive 中为方法调用时自动产生表示调用的返回结果的对象。ProActive 采用一种 Wait - by - necessity 方式来解决内部对象的同步,其思想如下:生成 Future 对象后可继续往下执行,除非是直接对 Future 对象的引用,才会自动停下等待,直到 Future 对象得到具体数值。Future 对象的值变为可用时,会自动得到更新。

1.3 节点的部署

开发分布式应用程序的时候,计算节点的部署往往是比较麻烦的[5]。ProActive 开发包提供了强大的 XML 部署描述器,可方便地开发实现计算节点的部署。ProActive 的部署文件是一个 XML 类型文件,它主要由三个部分构成:componentDefinition、deployment 及 infrastructure,用来提供虚拟节点(VirtualNode,简称 VN)、Java 虚拟机(JVM)及节点(Node)的映射关系信息。ProActive 在程序运行时从部署文件获取节点部署信息。有关详细概念可参见文献[4]。

2 P - Spider2.0 的设计与实现

2.1 P - Spider2.0 的系统框架

P - Spider2.0 整个系统分为 5 个层次,它们的结构如图 1 所示。

P - Spider Application(P - Spider 应用程序)
ProActive(Java 并行计算开发包)
JVM(提供 Java 环境)
SSH(连接协议)
PC 网络(LAN、Cluster)

图 1 P - Spider2.0 的层次结构图

第一层:物理层,由高速互联的 PC 工作站构成,可以是 LAN 或 Cluster。提供平台所需的物理设备,包括网络联接设备和 PC 等,它们提供了执行计算程序的环境。

第二层:连接层,采用 SSH 作为连接协议,为计算节点的动态部署提供安全的部署连接。

第三层:JVM 层,为 ProActive 的通信及运行提供 java 环境支持。由于 ProActive 是一个纯 java 的中间件,必须要有 JVM 为其提供运行环境支持,同时也为系统的跨平台提供支持。

第四层:并行软件层,采用 ProActive 并行分布开发包,支持平台及应用程序的开发。相当于一般高性能计算系统中的 MPI/PVM 等并行计算开发包的功能,以支持高性能计算。

第五层:应用层,基于 ProActive 开发的能够在该平台下运行的分布式并行 Web Spider 应用程序。

P - Spider2.0 采用对等节点的分布式并行设计方案。整个系统没有中心节点,各个节点完全对等,都是既作为协调器(SpiderCoordinator),又作为爬行器(SpiderWorker)。各个节点通过高速局域网进行通信。

P - Spider2.0 的协调器和爬行器都被设计成主动对象。协调器部分负责管理和维护 URL 队列的一个子队列,其中包括两个 URL 队列和 URL 去重器等模块。爬行器(SpiderWorker)部分负责网页采集、分析和报告发现的 URL 等工作。爬行器包括下载模块、DNS 解析器、URL 的抽取器、过滤器和分发器等模块。由于系统采用了我们设计的 RP 算法,RP 算法是基于 Rabin 算法的,需要计算每个 URL 的指纹,所以在 SpiderWorker 中有一个 Rabin 指纹生成器。

2.2 协调器

P - Spider2.0 的每个节点有一个协调器主动对象。每个协调器中有一个待采集 URL 的队列和一个 URL 检索模块。此待采集 URL 队列是为本节点的爬行器提供要采集的 URL。URL 检索模块采用我们自己设计的算法对 URL 进行去除重复。由于此算法是基于 Rabin 算法的,又是为 P - Spider2.0 系统设计的,所以称其为 Rabin P - Spider 算法,简称 RP 算法。

2.2.1 RP 算法

算法的基本思想是,首先构造向量 $(x_0, x_1, \cdots,$

$x_i,\cdots,x_n)$,$x_i = 0,1$,其中 i 为整数且 $i \in [0,\max(\text{rabincode})]$,初始值都设置为 0,然后用 Rabin 指纹算法[6]计算 URL 的指纹,并将得到的二进制序列映射成整数 i,最后用 i 作为向量的下标判断 x_i 的值,如果 $x_i = 0$,则此 URL 未被访问过,将 x_i 的值设置为 1;如果 $x_i = 1$,则此 URL 已经被访问过,将其丢弃。

算法描述如下:

```
Boolean RP(String url){
/*输入URL字符串,如果URL已经出现过则输出true,如果未出现过则输出false*/
unsigned int rabincode;
BitSet rabinBitSet1 = new BitSet(MAXRABINCODE);
//定义位向量,默认初值为false
rabincode = rabinfunction(url);
//计算机URL的Rabin指纹
if(false = = rabinBitSet.get(rabincode)){
rabinBitSet1.set(rabincode);
//将rabincode对应的位置true
return false;
}
return true;
}
```

其中 BitSet 类实现了一个按需增长的位向量。位 set 的每个组件都有一个 boolean 值。用非负的整数将 BitSet 的位编入索引。可以对每个编入索引的位进行测试、设置或者清除。

此算法,记录一条 URL 只需一位,判断 URL 是否已经访问过时,只需用索引寻址,即基址加上偏移量,对相应的位的状态进行判断即可。这样既简化了检索的过程,提高了检索速度,又有效地节省了存储空间。用从网上采集得到的 4500 万条 URL 对此算法与天网的 Hflp 算法做了去重速度对比实验。结果如表 1 所示。

表 1　RP 算法与 Hflp 算法对比实验结果

处理 URL 数(万条)	2500	3000	3500	4000	4500
Hflp 算法(万条/秒)	29.74	32.34	28.56	27.05	24.00
RP 算法(万条/秒)	45.13	43.13	41.31	40.68	39.49

实验数据表明,对 URL 的检索 RP 算法的速度明显比用 Hflp 作为 hash 函数的 hash 算法快。这是因为 RP 算法简化了去重过程,避免了过多的字符直接比较。

2.2.2　协调器的 URL 处理策略

根据节点的个数 m 把 RP 算法中的位向量$(x_0, x_1,\cdots,x_i,\cdots,x_n)$分成 m 段,如果是同构的计算机,可以平均成 m 段,如果是异构的计算机,可以根据计算能力分成不等的 m 段。每个节点的协调器根据相应的段创建一个位向量。这 m 个位向量构成向量$(x_0,x_1,\cdots,x_i,\cdots,x_n)$的一个划分。这里对向量$(x_0,x_1,\cdots,x_i,\cdots,x_n)$的划分也就是对下载任务的划分。

这样的策略下,URL 被根据指纹分配到各个节点上,爬行器只从本节点协调器的 URL 队列读取要采集的 URL。这样每条 URL 至多经过一次节点间传递(如果此 URL 属于本节点范围,则为零次)。如图 2 所示,例如节点 i 发现了一个 URL,记为 s,然后用 Rabin 算法计算 s 的指纹,并将其映射成为整数记为 k,如果 RP 算法的向量 x_k 属于节点 i 的向量区间,那么 s 被发送到节点 i 的 URL 去重器,去重结果如果 s 是新的 URL 则加入到节点 i 的待采集 URL 队列中,等待节点 i 的下载模块采集,这样 s 就不需要经过网络传递了。如果 x_k 属于节点 $j(j \neq i)$ 的向量区间,则将 s 发送到节点 j,节点 j 的 URL 去重器经过去重,如果 s 为新的 URL 则加入节点 j 的待采集 URL 队列,等待节点 j 的下载模块采集,这样 s 就经过了一次网络传递。

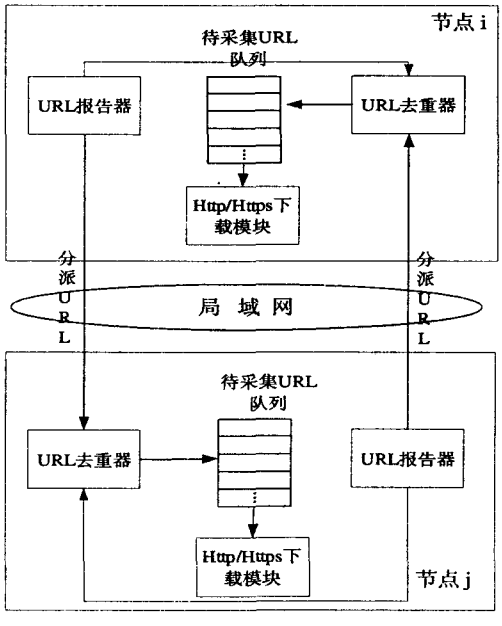

图 2　P-Spider2.0 URL 流程图

2.3　爬行器

每个节点上部署多个 SpiderWorker 主动对象。SpiderWorker 根据从本节点 URL 队列读取到的 URL 下载页面,然后解析页面的 HTML,提取出其中包含的 URL。将提取出来的 URL 链接按照预先定义的统一的格式补充完整。对这些 URL 进行过滤,然后用 Rabin 算法计算 URL 的指纹。最后,根据指纹判断应该向哪个节点的协调器报告发现的 URL 并发送。

这样的结构策略下,待采集的 URL 都是从本节点的队列中读取的,要比中心节点的通过网络分派速度快得多。各个 SpiderWorker 向不同的节点报告 URL,

能有效避免中心节点集中的网络传输压力。

由于主动对象具有的特性,对这些主动对象的方法调用,在形式上和对普通对象的方法调用是一致的。在调用的时候根本不用考虑对象在哪台计算机,哪个 JVM 和哪个 Node 上。设置好服务规则(默认是 FIFO)后,也不用考虑具体的实现细节。因为主动对象提供服务是根据规则有序的,所以也不必考虑同步问题。

2.4 P-Spider2.0 主要类的实现

P-Spider2.0 的 SpiderWorker 类主要算法如下:

```
public class SpiderWorker {
public IntWrapper startwork(){
int count=0;//统计下载的 URL 数
while ((curUrl = dequeue(urlQueue))! =null) par-do{
//各个节点并行工作
page=downloadPage(formatUrl(curUrl));
foundUrls=extractUrls(page);//发现页面中 URL
rabincode=rabinfunction(foundUrls);
//用 Rabin 算法计算 URL 的指纹
reportUrl(foundUrls, rabincode);
//根据指纹向不同的协调器报告发现的 URL
count++;
}
return new IntWrapper(count);
//ProActive 异步调用条件,要求返回包装类型
}
}
```

2.5 P-Spider2.0 系统的启动

整个 Web 可以看作是一张有向图 $G=(V, E)$ 组成, V 表示网页的 URL, E 表示两个网页之间存在的超链接 URL, 即一个网页中有另一个网页的 URL[7]。对于图中任意两个顶点 $V_i, V_j \in V$, 如果 V_i 到 V_j 有路径,则称 V_i 与 V_j 是连通的。假设存在集合 V_s, 其中初始仅起始 URL, 随着对 G 的遍历,不断地扩充 V_s, 对于 G 中任意一个 $V_i \in V$, 存在 $V_{si} \in V_s$, 从 V_{si} 到 V_i 有路径,则认为 G 是连通的。所以 Web 的搜集过程可以看作是从集合 V_s 出发,发现有向图 G 中所有 V 的过程。为了尽快发现有向图 G 中所有的 V, 应该采用系统从多个起始 URL 开始。所以启动 P-Spider2.0 系统的时候,先分配多个 URL 到各个节点作为种子 URL。

3 实验及分析

P-Spider2.0 在实现中引用了 Jeff Heaton 的 bot 包, bot 包是单机环境下的一个多线程的 Web Spider。为了验证 P-Spider2.0 系统架构的有效性和分布并行的效果,用单机多线程的 Spider 和 P-Spider2.0 做了对比实验。

实验环境如下:4 台 CPU 为 Pentium4 2.4GHz,内存 512MB 的计算机,通过百兆局域网链接到 Internet。软件环境为 Linux Red Hat9、JDK1.5 和 ProActive3.1。

首先,使用一台计算机,对 Internet 做不限制范围的采集 10 小时,记录采集的数据文件总量、下载的 URL 条数,计算出采集数据的速率和下载 URL 的速率。实验程序使用 bot 包中单机多线程的 Web Spider。

然后,使用 4 台计算机,同样对 Internet 做不限制范围的采集 10 小时,记录采集的数据文件总量、下载的 URL 条数,计算出采集数据的速率和下载 URL 的速率。P-Spider2.0 与单机 Spider 对比实验结果如表 2 所示。

表 2 P-Spider2.0 与单机 Spider 对比实验结果

实验系统	计算机台数	下载 URL 速度(条/s)	下载数据速度(k/s)
单机 Spider	1	4.56	107.66
P-Spider2.0	4	13.45	324.15

从实验结果可以看出,单机 Spider 的采集效率远不如分布式并行的 P-Spider2.0。这是因为,单机多线程只是并发程序,并不是真正的并行,单机 CPU 的计算能力和内存等系统资源也十分有限,而且同步方法和一些独占资源也成了效率的瓶颈。

分布式并行的 P-Spider2.0 的测试采集效率之所以比单机多线程的 Spider 有显著的提高,主要有四个方面的原因:第一是得益于处理机和网络接口数量的增加;第二是得益于基于 ProActive 的节点对等系统架构,有效地解决了中心节点瓶颈,减小了系统内耗;第三是在 URL 去重上采用了更加高效的去重算法 RP 算法;第四是主动对象这种用 Future 对象来实现的异步调用机制在一定程度上减少了线程的等待,提高了 P-Spider2.0 的采集效率。

4 结束语

结合 ProActive 中间件的特点[8]介绍了由我们设计开发的分布式并行的 P-Spider2.0 系统。ProActive 使得 P-Spider2.0 的设计和开发更加简洁、方便、灵活,大大降低了设计开发的代价。节点对等的架构去除了中心节点瓶颈,内耗更小,负载更均匀。RP 算法使 URL 去重效率更高。实验表明,基于 ProActive 的 P-Spider2.0 对 Web Spider 的采集效率有较大提高,整体架构简洁有效,可扩展性良好。

(下转第 202 页)

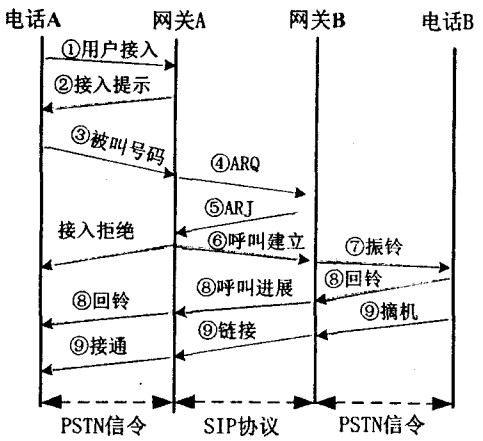

图 4 呼叫建立过程

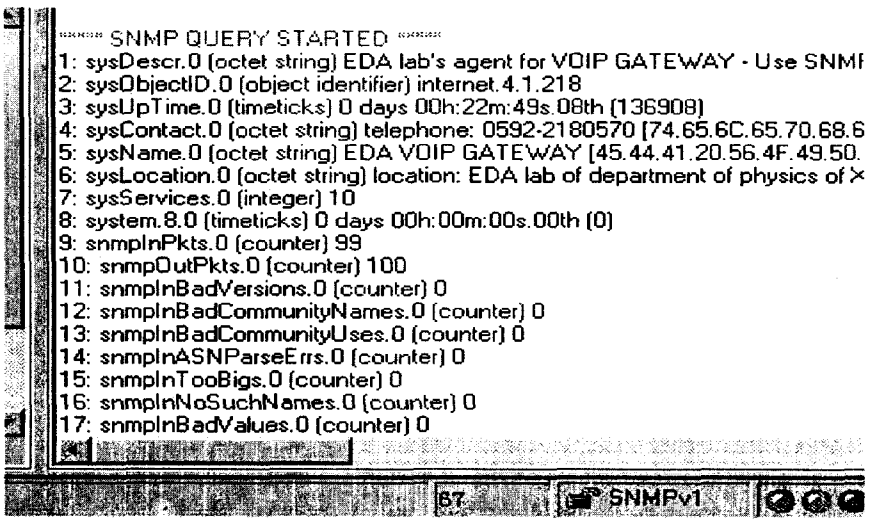

图 5 SNMP System 组的测试信息

4 结束语

通过 IP 网络传输语音包可以极大地节省带宽，并有助于传统 PSTN 电话网与 IP 网络的融合，IP 语音网关实现了 PSTN 与 IP 网络的互通，是开展 VoIP 业务关键设备。在对IP语音网关的功能进行分析的基础上，提出一个 IP 语音网关的设计方案，并采用现有的硬件条件及软件开发包加以实现，可为企业级的 VoIP 业务提供平台，并采用 SNMP 协议，实现对网关的远程监控管理。实践表明该方案具有使用方便、成本低等特点。

参考文献：

[1] 糜正琨. IP 网络电话技术[M]. 北京：人民邮电出版社，2002.

[2] Black U. IP 语音技术[M]. 北京：机械工业出版社，2000.

[3] 宋铁成，张 华. IP 电话的关键技术及其应用[J]. 电信技术，2001(2)：22－23.

[4] 林晓鹏，吕迎阳，郭东辉. 基于 Internet 语音通信的技术问题分析[J]. 计算机工程与应用，2002(8)：159－162.

[5] H.323：Packet－base multimedia communications systems[S/OL]. 1996. http://www.itu.int/rec/T－REC－H.323/.

[6] RFC3261：Session Initiation Protocol (SIP) [S/OL]. 2002. http://www.ietf.org/rfc/rfc3261.txt.

[7] 郑红英，郭东辉，纪安妮，等. 语音压缩技术及其应用的进展[J]. 计算机与网络，2000(5)：27－29.

[8] MySQL4.1.22[CP/OL]. 2004. http:// www.mysql.com/.

[9] RFC1157：Simple Net Management Protocol(SNMP)[S/OL]. 1998. http://www.ietf.org/rfc/rfc1157.txt.

[10] SNMP++[CP/OL]. 2006. http://www.agentpp.com/snmp_pp3_x/snmp_pp3_x.html.

(上接第 198 页)

参考文献：

[1] Zeinalipour－Yazti D, Dikaiakos M. Design and Implementation of a Distributed Crawler and Filtering Processor[C]// Next Generation Information Technologies and Systems. Berlin: Springer, 2004:149－150.

[2] Dongarra J. Sourcebook of parallel Computing[M]. Chicago: Morgan Kaufmann Publishers, 2002.

[3] Baude F, Caromel D, Morel M. From Distributed Objects to Hierarchical Grid Components[M]// Lecture Notes in Computer Science, 2003:1226－1242.

[4] Huet F, Caromel D, Bal H E. A High Performance Java Middleware with a Real Application[C]// Supercomputing, 2004. Proceedings of the ACM/IEEE SC2004 Conference. Pittsburgh, Pennsylvania: IEEE Computer Society, 2004.

[5] Baduel L, Baude F, Caromel D, et al. Programming Composing Deploying for the Grid[M]// Grid Computing: Software Environments and Tools. London: Springer, 2007:205－229.

[6] Broder A Z. Some applications of Rabin's fingerprinting method[M]. New York, NY: Springer－Verlag, 1993:143－152.

[7] 王小林，刘宏申. 搜索引擎的设计研究[J]. 计算机技术与发展，2007，17(2)：5－7.

[8] 董明刚，梁正友. Windows 下基于 ProActive 并行计算的关键技术[J]. 计算机工程，2006，32(19)：105－107.