维普资讯 http://www.cqvip.com

离线浏览下载引擎算法的优化及改进*)

Optimization and Refinement of the Algorithm of an Offline Browser Download Engine

吴 晓 刘君悦 孙玉芳 李 彤

(中国科学院软件研究所 北京100080)

Abstract The refined algorithm is put forward mainly based on the Wget. Combining the advantages of HTTrack and Wget, the download algorithm is rewritten on the following aspects: the download process is changed from depth-first retrieval to width-first retrieval; from single-socket to multi-socket; and from recursive retrieval to loop retrieval. The original process, which includes the system calls that maybe cause the network blocking, is cutted into different sectors, and runs in the way of multitask. The results of experiments and feedbacks from actual application show that the refined algorithm gets great improvement on the download efficiency and memory occupancy.

Keywords Download engine, Recursive retrieval, Polling, Blocking I/O

1. 前言

针对政府、军队等机要部门信息安全方面的需求,我们开发了基于 Linux/Unix 平台的"内外网信息安全控制系统"(参见图1),通过采用中间服务器和安全电子开关,实现内部局域网和 Internet 的分离,在保证任何时刻内部局域网和外网均处于断路状态的前提下,通过采集器机群将外网信息选择性地采集到内网,整个过程信息均单向传输,从根本上保证了内部局域网系统的安全性,同时实现了对外网资源的可控采集和访问。

执行采集任务的下载引擎是本系统的核心,扮演着极其重要的角色,它的效率直接影响着整个系统的性能。目前比较流行的离线浏览下载引擎有:HTTrack, Wget, Pavuk, Teleport 等,而这些程序都不同程度地存在一定问题,且下载效率不高,在此基础上,我们结合各自的优点,将其算法进行了修改,重写了程序,取得了较好的效果。本系统在某些部门使用过程中收到良好效果,很受用户欢迎。

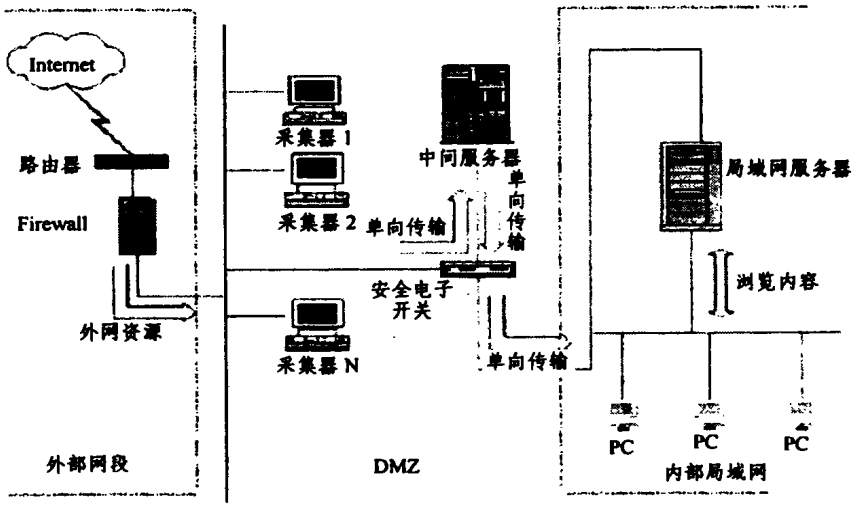


图1 内外网信息安全控制系统示意图

2. 下载引擎对比分析

2.1 HTTrack

HTTrack[3]是一个容易使用的站点镜像软件,能将网站传送到本地目录,并且会把所有相关链接重新组织,以便顺利地离线浏览。HTTrack 可以更新一个已经存在的站点,支持断点续传,多连接模式,最大化地提高下载速度。然而,HTTrack 存在着以下几个显著的缺点:1)占用的内存较大;2)经常性的系统崩溃;3)多线程机制不稳定;4)编程结构不良,模块化不明显,源代码风格差,不易功能扩展。

·HTTrack 算法分析

(下载引擎可以按 FTP 及 HTTP 方式进行下载,在此,以 HTTP 方式进行介绍。)

```
while (getQueue( )) /* 从队列中取出链接进行处理 */
{  back_loop( ); /* 任务根据状态标志进行流水处理; */
   if (is_html( )) /* 下载完成页面如果是 HTML/TEXT 类型 */
      enQueue( ); /* 页面中的链接插入到队列中 */
}
```

HTTrack 按照广度优先搜索算法,采用了流水线机制,建立了 HASH 表,通过 lien_back 结构中的 status 状态标记,

*)本文得到国家自然科学基金(60073022)、国家863高科技项目(863-306-ZD12-14-2)、中科院知识创新基金(KGCX1-09)等项目的资助。

维普资讯 http://www.cqvip.com

配合循环结构使得多个任务同时运行。下载的主要结果分别记录于三个文件中:new.dat(保存所有下载的 HTML 文件)、new.ndx(记录各文件在 dat 中的索引地址)、new.lst(记录已下载的文件名)。

2.2 Wget

Wget[4]是一个从 WWW 上用 HTTP 和 FTP 两种协议方式下载文件的自由软件。Wget 有很多有用的特性使得下载更易进行,而且程序风格良好,结构清晰。它是以非交互的方式运行,因而可以在后台运行。能够根据 HTML 的文档结构或 FTP 的目录树递归地下载文件。Wget 在网速比较慢或者网络连接不稳定的时候,表现比较良好。它将不断地重试下载直到完全下载或者达到最大的重试次数。然而,也正因为这些特点,使得 Wget 表现并没有达到我们的期望。其缺点是:1)没有 HASH 机制,在处理较多文件时速度大大降低;2)简单的递归,导致内存消耗较多;3)深度优先递归搜索,下载结果树不平衡,下载效果不理想;4)没有使用多 socket,使得下载效率较低;5)当网络速度较慢时,不会及时跳出,会出现长时间等待。

·Wget 算法分析

```
recursive_retrieve() {
    while (1) {
        parseusrl();
        retrieve_url();
        get_urls_html();
        for (页面中的链接) {
          if (html)
            recursive_retrieve();
        }
        convert_links();
    }
}
```

Wget 是一个深度优先的递归程序。给定一个 URL,首先对其 HTML 进行分析,之后建立网络连接,先解析文件的头部,然后下载 HTML 文档对应的相关文件,对于每一个下载完成的文件,获取其页面中的链接,之后将页面中的链接转换为相对链接。如果新下载的文档也是 text/html 类型,则递归地进行页面下载,如此反复。

3. 算法的优化及改进

3.1 套接字读写操作的 I/O 模式

套接字的读写操作缺省情况下使用的是阻塞 I/O 方式。执行读写 I/O 的系统调用时,执行或等待 I/O 操作的进程挂起,即进程阻塞,直到 I/O 操作完成,调用才返回并唤醒进程继续向下执行。当采用阻塞 I/O 方式的套接字读写操作时,如果网络速度较慢或者网络连接很不稳定,数据未能及时到达,则进程将无限制等待,不能继续执行下面的程序,大量的时间将用于等待 I/O 操作上。

如果套接字的读写操作采用选择等待 I/O 模式,通过系统调用 select()实现,当没有设备准备好时,select()挂起,其中任一设备准备好,select()就返回。进程可以对准备好的这个设备进行操作。程序中调用可能造成阻塞的函数时,如果发生阻塞,则这些函数返回,程序可继续向下运行。当可能阻塞的函数所对应的任务完成时,则当程序再次调用该函数时,就返回零表示运行结束。

非阻塞模式的套接字读写操作可以避免程序死锁,但是需要程序不断检查各个可能阻塞函数的状态,并需要使用一个循环来不断地测试某个文件描述符是否有数据可读(称作 polling)。应用程序不停地 polling 内核来检查是否 I/O 操作已经就绪。一般情况下,套接字的读写操作采取非阻塞模式与同步 I/O 模式组合使用。

I/O 多路复用即同步 I/O 模式,使用 select()、poll()等函数实现对多个套接字的同步 I/O 操作。它能同时等待多个套接字描述符,而这些套接字描述符其中的任何一个进入读/写就绪/出错状态时,select()函数就可以返回。

3.2 算法优化策略

我们在比较了以上两个下载引擎优缺点的情况下,主要以 Wget 为基础,结合各自的优点,重写了算法,将下载引擎算法从以下几方面进行了优化及改进。

3.2.1 *多任务并发,多套接字* 当只有一个套接字时,如果网络阻塞时,系统将长时间得不到所需数据而处于等待中,不能进行其它的操作。进程为等待某个 I/O 操作而阻塞,而其他 I/O 已经准备好,却不能得到及时的响应,导致 CPU 使用效率很低,资源分配很不合理。因此,为了提高程序的效率,加快下载的速度,我们将任务数由一个增加到了八个,把单套接字变为多套接字,使一个进程同时处理多个文件描述符,多个任务并发执行。这样,当某个任务所需的数据准备好之后,即可对其进行读写,若所需的数据未准备好时,它可以进行其它步骤的操作,而不会处于阻塞状态,也不会因某个任务未得到数据而影响其它任务的运行。

3.2.2 *广度优先搜索* 当采用深度优先搜索时,系统将不断地搜索链接,直到限定的层数上,而对于用户而言,如果能尽可能多地浏览到某层数上的页面则可能较为理想,而且深度优先搜索,下载结果树不平衡。因此,我们将深度优先搜索改为了广度优先搜索,把所需下载的链接信息放入一个队列,每个任务从这队列中取出一个链接进行下载,当一个 text/html 类型的页面下载完成后,将其页面中所包含的链接(主要是通过识别 href 和 src 标记)加入到队列中。当队列中没有链接时,程序将终止运行。

3.2.3 *轮询机制* 我们把 Wget 下载引擎算法的递归及其函数调用根据需要进行了完全调整,将与网络有关的操作分离出来形成单独的步骤,抛弃系统原来所采用的阻塞 I/O 方式,而采用非阻塞 I/O 方式,系统将不断地轮询各个任务的状态,分别对各个任务进行处理,当某一任务当前步骤处理完成,则该任务状态改变,系统继续处理另一任务,而当存在网络阻塞时,系统能及时返回,转而处理另一任务,而不是盲目等待,节省了时间与资源,提高了效率。

3.2.4 *I/O 多路复用* 程序在每次循环开始时,都预先读取信息,以供后面的处理使用。在此,采用了 I/O 多路复用方式,多个套接字同时获取信息,一旦信息可读就将数据提前放入到各个任务相应的缓冲区中以供程序处理。

3.2.5 *二级队列机制* 如果采用一般广度优先搜索的队列,所有的 URL 链接信息都保存在一个 URL 队列中,当下载文件数达上千个之后,与之相关的链接数也随之增加,使得存储链接信息的 URL 队列占用的内存不断增大,达到几十兆,甚至上百兆,而且,随着程序的运行,将按多项式增长。我们采用了二级队列机制(见图2),增加了一个文件队列,用于存放已下载文件的信息,作为一级队列,程序从文件队列中取出一个文件进行分析,将分析得到的链接放入 URL 队列(二级队列)中,从 URL 队列中读取链接进行下载,当 URL 队列为空时,即完成一个页面相关链接的下载后,就从文件队列中再读取一个文件进行分析,将分析得到的链接加入到 URL 队列中,如此反复。这样,URL 链接信息所占用的内存

维普资讯 http://www.cqvip.com

只是一个文件所包含的 URL 队列以及已下载文件队列的空间，而不必同时保存所有文件的所有链接。采用二级队列机制后，可保证系统占用的内存基本恒定。

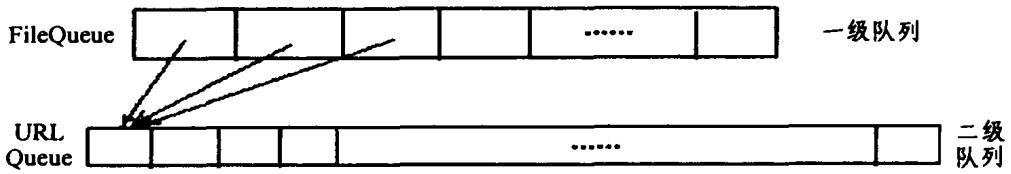


图2 二级队列示意图

3.3 New_wget 算法设计

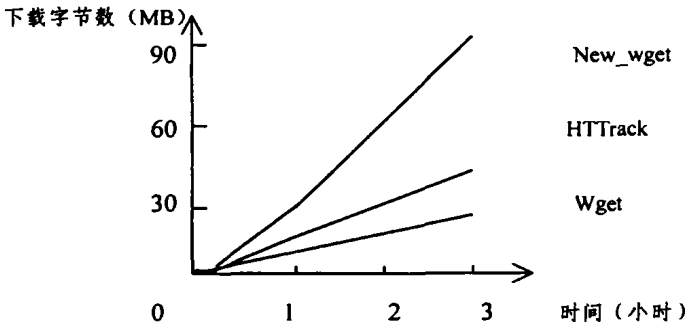
新的下载算法将下载过程由深度优先转换为广度优先，由单套接字变为多套接字，增加了 HASH 表，并将递归调用转换为循环分步调用，将可能存在网络阻塞的部分分为不同的阶段，多个任务同时运行，当某一任务出现阻塞时，能及时跳出本次循环，而让其它任务继续执行。而当该任务某一步骤完成后，状态改变，执行下一步骤。如此反复执行，直到符合某个条件，如达到一定下载文件数量、到达一定的下载层次等，程序才退出。以下是改进后的 New_wget 主要算法框架：

```
new_initialize( ); /* 系统初始化; */
while(!task_list){
    new_read(); /* 将下载数据预读到缓冲区; */
        /* 该函数实现了 I/O 多数复用及多套接字机制 */
    for(每一个任务) { /* 通过循环及以下步骤配合实现了多任务
        并发及轮询机制 */
        switch(任务状态) { /* 通过 case 0 和 20实现了广度优先搜
                索 */
        case 0: new_get_one_task_from_list();/* 从队列中读取
            一个 URL 以供处理; */
        case 10: new_parse_url(); /* 解析 URL; */
        case 20: new_getproxy(); /* 获取代理服务器信息; */
        case 30: new_url_file_name(); /* 读取 URL,取得存放于
            本地的路径及名称信息; */
        case 40: new_connection(); /* 建立网络连接; */
        case 50: new_sendrequest(); /* 向服务器发送请求; */
        case 60: new_header_get(); /* 接收文件头信息; */
        case 70: new_parse_header(); /* 解析头,获取下载文档信
            息; */
        case 80: new_get_contents(); /* 获取文件信息,并保存到
            本地; */
        case 90: new_get_urls_html_enlist(); /* 从 html 文档中
            获取页面中的链接加入队列; */
        case 100:new_convert_link(); /* 将页面链接接换为本地
            链接; */
        case 110:new_free_task(); /* 将处理完成的链接从队列
            中删除; */
        }
    }
}
```

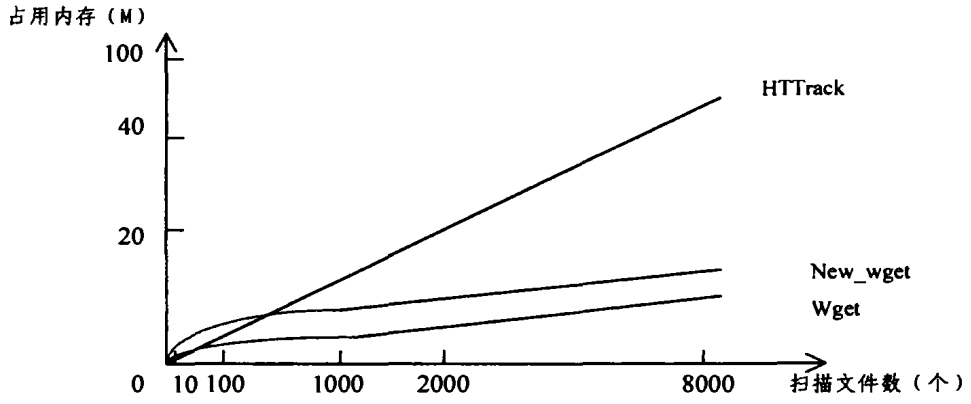
4. 性能比较与展望

New_wget 由于采用了多套接字、多任务并发、非阻塞式 I/O、I/O 多路复用方式以及轮询机制，修改了原下载算法，在下载效率及内存占用等方面取得较为理想的效果。虽然系统初始化时，在建立多任务的过程中会比其它程序稍慢，但随着程序的运行，下载文件数达到十个以上时，程序的下载效率显著提高，尤其是网络带宽一般，或是对方网络响应速度较慢，程序长时间运行时，其下载效率、内存占用、稳定性、可靠性、健壮性等方面的优势就更为明显。

（下转第117页）



(a) 程序下载效率比较



(b) 程序内存占用情况比较

图3 效率及内存占用情况比较

维普资讯 http://www.cqvip.com

中，无需另外设计数据存储模型。在 Bullant 中，客户端的用户界面是在网络服务器上使用 GUI 类库创建的，无需为客户端的界面显示专门编写程序。因此，使用 Bullant 的 Web 应用开发是在网络服务器上一体化的环境中完成的，简化了应用内部结构的层次，降低了系统设计的复杂度。

在 Web 应用开发的初期，由于 Bullant 网络服务器中需要使用 ACID 内存，该服务器的硬件配置要求优于其它解决方案中的服务器，然而也是因为使用 ACID 内存，无需购买有关数据库管理系统的软件，可以显著降低在软件方面的投资。Bullant 网络服务器具有线性可扩展性，当 Web 应用的用户数目上升时，可以通过单纯增加服务器中 CPU 的数目以及扩充内存来提高 Web 服务的性能，无需对应用程序做任何改动。在其它解决方案中，往往是通过增加服务器的数目来提高 Web 服务的性能，在多服务器的情况下，产生了服务器之间的通讯连接、任务在各个服务器之间的分配等问题。为了维持系统运行的连贯性，通常需要对应用程序进行修改，不仅要在硬件方面追加投资，还要承担软件维护的费用。因此，从系统的整体运营成本方面考虑，Bullant 是一种更为经济的解决方案，可以最大限度地保护投资。但是，在系统开发之初必须考虑服务器配置的可扩展性，比如采用单 CPU 时，需要上千兆的内存，并且服务器需要采用能够添加 CPU 的体系结构，所以 Bullant 解决方案的初期硬件投资比较高，这可能是一个遗憾。

在 Bullant 解决方案中，客户端与网络服务器进行通讯所需带宽最小仅为9.6K，接入方式既可以是有线的，也可以是无线的，支持 PC、PDA、手机等多种接入设备。对于同一个应用，网络服务器根据不同接入设备的图形显示能力定义不同的用户界面的表现形式，用户可以利用多种接入设备作为客户端，随时随地与 Bullant 提供的 Web 应用进行交互。因此，采用 Bullant 解决方案的 Web 应用具有良好的可用性。

Bullant 应用中需要长期保留的数据保存在网络服务器的 Epoch 文件中，这些 Epoch 文件仅供系统使用，服务器中不存在数据库管理系统中那样的对外部开放的接口，所以这些数据无法被恶意窃取，即使攻击服务器得到了 Epoch 文件，其中的数据也是按照内部的格式组织的，无法解读。客户端在显示 Web 应用的用户界面时无需下载任何 Applet 或者脚本，从而避免了运行不安全代码的可能。同时，Bullant 服务器与客户端之间的专用协议可以与安全传输的标准协议 TLS 1.0协同工作，也支持使用128位的 RSA 算法进行加密传输。因此，Bullant 解决方案具有相当可靠的安全性。

结束语 本文介绍了一种新型的 Web 应用解决方案——Bullant，并将它同现有的其它解决方案进行了比较分析。Bullant 的 Web 应用解决方案中的 ACID 内存、支持多种接入方式与接入设备的对等远端以及建立在零摩擦引擎基础上的线性可扩展性，为 Web 应用的开发提供了一些新颖的思路。虽然开发中使用 Bullant 语言有别于目前流行的 C、C^{++} 或者 JAVA 等语言，但它是一种完全面向对象的程序设计语言，与 C^{++}，尤其是 JAVA 的编程方法极为相似，熟悉 JAVA 的程序设计人员甚至可以立即使用 Bullant 语言进行开发。采用 Bullant 的解决方案可以将用户界面、业务逻辑与永久数据相结合，对 Web 应用进行一体化描述，大大降低了应用内部层次的复杂度，使系统具有良好的扩展性。笔者所在的北京大学信息科学中心使用 Bullant 技术已经成功地开发了一个网上股票交易系统原型。相信 Bullant 技术会为 Web 应用的构建提供一个简洁、高效的解决方案。

参 考 文 献

1 Chong J. Real Business Processing Meets the Web, SIGMOD'98, Seattle, WA. USA, 1998.536

2 IBM Corporation. Integrating Data and Transactions for Agile e-Business, Aug. 2001. http://www-4.ibm.com/software/web-servers/appserv/whitepapers/wp_was_overview.pdf.

3 Microsoft Corporation. Introducing the Windows .NET Server Family, Nov. 2001. http://www.microsoft.com/windows.net-server/evaluation/overview/default.asp.

4 Teta srl Internet Consultant, StWeb Stratos Web and Application Server, Jan. 2002. http://www.stweb.org.

5 Serain D. Middleware, Springer, 1999

6 Bullant Technology Pty Ltd. CAN 069011114, bullant Technology White Paper, Jan. 2000

（上接第105页）

我们用四台采集器构成的机群分别用 HTTrack、Wget 以及我们改进后的 New_wget，在保证大致相同网络带宽及负载情况下，对本地局域网以及 www.sina.com.cn、www.peopledaily.com.cn、www.redhat.com 等一些国内外著名网站进行下载，并用 New_wget 对不同网站进行下载，将整个过程进行详细记录。图3是进行横向和纵向综合分析比较后得出的下载效率及内存占用情况分析结果。

从图中可以看出，由于采用新的策略，New_wget 在下载效率方面比 Wget 提高了近五倍，而在内存占用方面约是 HTTrack 的十分之一。

然而，New_wget 下载引擎在做 DNS 查询时效率较差，需要多次网络查询，较费时间，而且系统调用 gethostbyname()时速度较慢。鉴于此，下一阶段，我们将需要在相应部分做进一步改进，使用 pthread 机制，优化 Cache 机制，使下载引擎能发挥出最大优势。

参 考 文 献

1 http://www.LinuxAid.com.cn;

2 http://www.redflag-linux.com;

3 http://www.httrack.com;

4 http://www.gnu.org;

5 Stevens W R. UNIX 环境高级编程. 机械工业出版社，2000

6 怀石工作室. Linux 上的 C 编程. 中国电力出版社，2000.

7 邹海明，余祥宣. 计算机算法基础. 华中理工大学出版社，1995