**分布式网络爬虫设计文档**

宋其翰 王 存

1. **功能分析**

本次作业完成一个简单的分布式网络爬虫的设计，实现功能包括：

1. 从给定网址作为入口，爬取对应入口网页并分析其中的URL并爬取对应的网页，直到没有更多页面爬取时完成。
2. 爬取时只爬取本地服务器的网页，不怕取外站资源。
3. 采用多线程结构设计，输出每一个文件的文件大小。
4. **架构设计**

本系统整体架构如下图1所示，由主线程、抓取线程、分析分配线程和分析线程五类线程组成。线程间的通信主要使用一个轻量级消息队列Nanomsg来进行。主线程主要负责抓取线程和分析分配线程的创建。抓取线程主要负责网页的Socket的抓取，并将得到的网页内容放到队列中。分析分配线程主要负责从消息队列中提取得到的网页内容，并将其分配到分析进程的进程池中对应的线程进行计算。分析线程主要负责从得到网页内容中解析出地址并放到消息队列中。

消息队列

分析分配线程

分析分配线程

分析分配线程

消息队列

网页抓取线程

消息队列

图1 架构图

1. **核心线程设计**

本系统的核心线程为网页抓取线程和网页分析线程两个。两个线程间的通讯主要由Nanomsg消息队列完成。

* 1. **网页抓取线程**

网页抓取线程采用的是Libevent库来进行网络的连接和抓取，Libevent库是一个基于Reactor模型的一个网络库，支持异步的调用函数。只需在初始化的时候在相应时间上一个函数，程序就会在该事件发生时调用对应的函数库。

1、Bufferevents和Evbuffer

Libevent在event事件调用的基础上，又在之上封装了一个Bufferevents，主要用于面向流的通讯，其类似Java的输入输出流，通过指定IP地址和端口，就能直接进行读写。在Bufferevents中封装了两个Evbuffer类型的指针，一个是input，另外一个是output。Evbuffer类型主要用于存储字符串内容，是一个能够用在网络编程的缓冲区，类似C++中的String类型。只要一次定义，可以无限的进行读取和写入而不用担心缓冲区溢出的问题。但是在bufferevent中，每次读取的字节数只能是4096个字节。对于大量的数据，这个缓冲区大小是比较小的。

2、HTTP的keep-alive

HTTP包头的keep-alive能够保证在每一次传输完成后保持连接可以继续下一次的数据传递，这样可以大大节省每次打开和关闭TCP连接的时间开销。但是keep-alive并不是无限次的，不同的HTTP服务器默认对keep-alive的请求次数和超时时间的设置是不一样的。Apache默认的keep-alive请求次数是100次，超时时间是5s。Nginx默认的请求次数也是100次，但是超时时间是75s。

在请求HTTP包的时候，其请求顺序是发送一个HTTP请求包，HTTP服务器就会返回相应的请求页面，待当前页面发送完成后，再接收下一个HTTP请求包。而不是能够一直接受HTTP请求包。

3、线程主要逻辑

在程序一开始，初始化Bufferevents，设置为在端口可读和有其他事件发生时调用相应的回调函数。当端口可读时，调用eventRead函数，在eventRead函数中，读取当前缓冲区的内容，并通过状态机判断并分析当前的包内容，一旦读取到一个完整的包后，如果当前包的HTTP状态是200，就将当前网页的内容和网页地址发送给链接分析线程进行分析，否则就直接丢弃该包。然后函数从消息队列中取出下一个要连接的网页地址，构造一个http请求包，写入到Bufferevents中。当有其他事件发生时，调用eventcb函数，该函数中主要处理socket连接上和socket结束的两个事件。当socket连接上时，就从消息队列中取出要取得的网页地址，并构造相应的HTTP包头，写入到对应的Bufferevents中。当socket断开时，释放当前的Bufferevents，重新构造一个新的Bufferevents并连接，就能够在HTTP服务器断开时，再次连接上服务器并继续取出相应的网页。

3、分析HTTP包头

在处理HTTP连接时，需要进行HTTP包头的分析。HTTP包头的处理也是一个状态机，其状态机的转换如下图2所示。

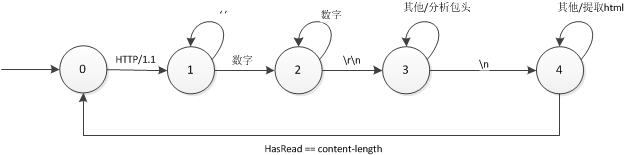


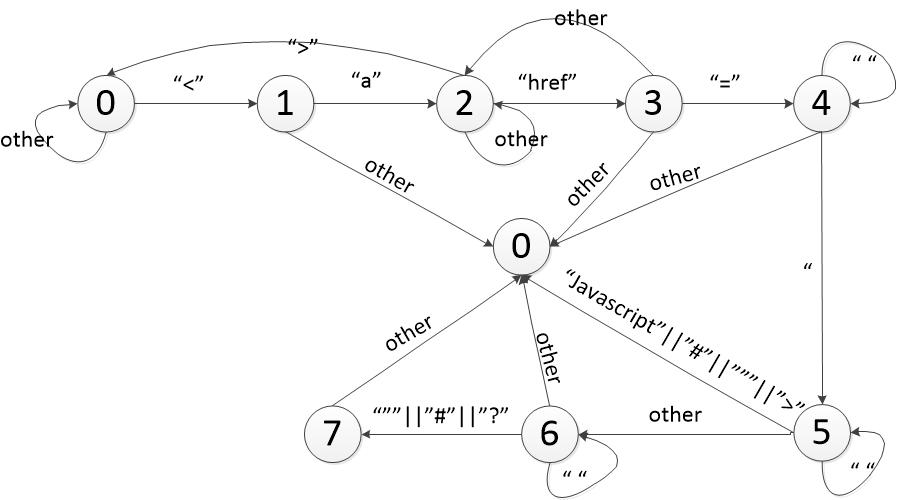
图2 HTTP包头处理的状态机

* 1. **分析线程**

分析线程将接收到的HTML语言，使用状态机的方法分析HTML中出现的链接，将外站的连接去除，只留下本地链接，并将本地链接中含有相对路径的部分转换成为绝对地址链接。

1. HTML中链接提取

HTML链接中的URL主要放在<a>标签中的href属性中，我们采用的状态机的方式提取出标签中的URL。分析时也需要去除href=”#”、href=”javascript::void(0)”等这样的无效链接，对于href=”a.html#a1”这样的锚点链接，我们要将后面的锚点去掉，只留下a.html。这样在去重的时候，能够保证不将a.html#a1和a.html#a2视为两个链接。本程序使用的状态机如下图3所示.

图3 HTML链接中提取HRL链接状态机

使用状态机的好处是简单易行，相比起解析HTML DOM模型能够更加高效的解析出URL地址，但是由于这个解析器不够完备，只能解析出<a>标签中href属性的地址。还有一些包含在<iframe>、<table> 标签backgroud属性等的网页链接地址，是不能够被这个状态机分析到的。

1. URL的转换

HTTP包的请求地址中是不能含有相对链接的，而且使用相对链接对于去重也是比较复杂的，所以我们需要将我们抓到的URL链接转换成为绝对链接，同时也要去除外站地址，避免我们的程序爬取到外站的链接导致程序无法收敛。

在转换相对链接是，需要转换诸如 ../../a.html、a.html、/a.html这样的链接，转换成为对应的相对地址。

1. URL地址的去重

为了避免重复的抓取相同的URL地址，在本地需要将URL去重。在本次实验中，使用的是实验一大规模字符串查找中的TRIE树来进行查重。在使用TRIE树的时候，需要注意进行线程的同步。由于上一个实验的代码是单线程的程序，所以TRIE树的插入、查找的操作也是单线程的。所以在使用时需要用pthread\_mutex\_t来进行线程间的同步与互斥操作。

* 1. **消息队列**

消息队列采用的是Nanomsg库，这是一个使用C语言编写的一个消息队列，可以轻松完成跨线程、跨进程、跨机器的通讯。Nanomsg的通讯模式有NN\_PAIR、NN\_REPREQ、NN\_PIPELINE等通讯模型。本程序采用的通讯模型主要是NN\_。PAIR的通讯模型，进行一对一的通讯。考虑到程序是在同一个进程内运行，所以我们主要以传递指针为主。下图4所示的就是两个进程通讯之间传递的数据类型：

typedef struct url\_req\_s{

char \*url;

struct evbuffer \*html;

}URL\_REQ;

URL\_REQ \*

网页抓取线程

网页分析线程

char \*

图4 两个进程之间传递的数据类型

从数据类型的定义中可以看出，本程序的所有的数据通讯都是以指针的来进行的。这样做节省了消息队列的空间，也避免了拷贝数据的时间影响。但是也造成了我们的程序不能够将分析线程和网页抓取线程分开在两个进程甚至机器之间进行通讯，从这点来看，这个程序并不能进行分布式的设计，这和题目的要求是不太符合的。

* 1. **线程池**

进程池线程主要采用的是threadpool.c和threadpool.h这两个文件。其中，create\_threadpool函数新建了一个线程池，并将线程池进行初始化操作。在实际需要调用线程的时候，只需要使用dispatch函数，将要调用的函数指针、参数传递进去。线程池会自动给你调用执行并回收资源。通过线程池可以很好的控制分析线程运行的个数，做到按需来分配线程，避免了因频繁生成和销毁线程造成的线程开销。

1. **实验结果**

经过多次运行,该程序总共分析出了115830个链接地址，有效链接地址为77262个网页。只是抓取了16W网页一半。猜测问题原因可能出在URL链接地址的分析模型上。自己的代码在程序实现的时候还是有一些 问题。

1. **存在问题**
2. 程序何时停止

由于时间问题，没有能具体到程序超时问题。所以程序一直在爬取，并不会结束。

1. 内存轻微泄露

由于释放原因，存在内存的轻微泄露。