一种基于 Ajax的爬虫模型的设计与实现

范轩苗[1] 郑 宁[1] 范 渊[2]
[1](杭州电子科技大学计算机学院 浙江 杭州 310018)
[2](杭州亚龙信息科技有限公司 浙江 杭州 310035)

摘 要 Ajax技术给传统的爬虫带来了巨大的挑战,为了能够提取出在 Ajax程序中隐藏的 WEB资源,需要解决两大关键问题:JavaScript脚本解析和每一次脚本事件触发后的页面状态信息的保存。提出一种适合 Ajax的爬虫模型,在模型中设计了一个嵌入的脚本解析引擎和使用了有向状态图,有效地解决了上述的两个关键问题。实验结果显示该爬虫模型能够有效地提取出 Ajax程序中的隐藏资源。

关键词 爬虫 脚本解析引擎 有向状态图

DESIGN AND IMPLEMENTATION OF A CRAWLER MODEL BASED ON AJAX

Fan Xuanmiao[1] Zheng Ning[1] Fan Yuan[2]
[1](School of Computer, Hangzhou Dianzi University, Hangzhou 310018, Zhejiang, China)
[2](Hangzhou DBAPP security Information Technology CO., LTD, Hangzhou 310035, Zhejiang, China)

Abstract Ajax technology brings to a great challenge on traditional crawler. In order to find the Hidden Web resource made by Ajax, two key problems have to be resolved: JavaScript parsing and storing every page's state information after every JavaScript's event triggered. The paper presents a crawler model suitable for Ajax, in which an embedded script-parsing engine and a directed state graphics have been designed, it has effectively settled above two problems. The experimental result shows that the crawl model could acquire the Hidden Web resource in the Ajax application effectively.

Keywords Crawler Script-parsing engine Directed status graphics

0 引 言

传统的 Web开发利用超链接和表单来定位资源,因此爬虫可以利用特定标记(例如标签 a)或者正则表达式静态地获取 Web资源。但是随着 Web技术的发展,一些 Web资源被隐藏了起来,这些资源被称为 Hidden-web[1],在提取隐藏资源面前,传统的爬虫已经变得无能为力。根据文献[4]提出的观点,可以将隐藏资源分为两种:基于服务端的资源和基于客户端的资源,第一种隐藏资源是针对隐藏在 form表单背后的资源,例如最常见的查询表单对应的数据库资源,对于该类资源的获取已有广泛的研究[1—4];第二类资源主要是针对由于客户端脚本动态操作 Web页面引起的,对于该类资源的获取,必须解决脚本解析问题,国内外研究提出了直接利用浏览器的 API[5]和 JavaScript脚本解析引擎—— SpiderMonkey提供的接口[6]来解决脚本解析问题,但是这些研究只是针对传统的 Web应用,并且在文献[6]中只考虑了脚本的部分解析。文献[7]提出了一种适合 Ajax的爬虫模型,解决了 Ajax通过异步请求初始加载内容的获取问题,但是没有支持事件触发,因此仍然不能进一步挖掘出整个 Ajax网站的资源。该文主要针对 Ajax的隐藏资源获取进行研究,提出了一种新的基于事件驱动的爬虫模型,能够较完整地提取出 Ajax的资源。

1 传统爬虫方式受到的挑战

传统爬虫在对于 Ajax的网站进行爬行受到巨大的挑战,主要体现在两个方面:一个是由于脚本使 Web资源不可见,另一个是由于 Ajax网站的自身结构特征导致,所有交互更新均发生在同一个页面上,无法有效地定位每一次更新后页面的状态信息。

1.1 Ajax资源的不可见性

Ajax利用 JavaScript绑定一切,包括各个元素的事件触发,使用 XMLHttpRequest对象与服务器进行异步通信,同时利用服务器返回的信息动态地对 DOM(文档对象模型)进行更新。这种动态的内容更新机制导致了一些资源对于传统爬虫来讲是透明的,主要体现在两方面:(1)异步请求;(2)异步响应。对使用了 Ajax技术的网页源码进行分析,发现在网站中没有传统的超链接资源,主要是通过网页中的 JavaScript脚本来动态地加载内容。对于仅靠超链接和表单来抓取资源的传统爬虫来说是不可能获取这些资源的。因为一个爬虫模型必须具备能够解析和执行 JavaScript脚本以及模拟事件触发的能力。

收稿日期:2008—09—06。浙江省自然科学基金(Y106176)。范轩苗,硕士生,主研领域:网络安全。

?1994-2015 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

1.2 状态信息的不可定位性

在传统的 Web应用程序中，用户的每一次请求都对应一个页面的响应，也就是说每一次更新后的状态都有一个 URL与之对应，每一个响应的页面实际上记录着每一次的状态信息。但是，在 Ajax应用程序中，每一个状态没有对应一个明确的 URL。在很多情况下，一个 Ajax应用程序只存在着一个页面，包含了许多入口点与用户交互，因此，无法利用 URL来定位每一个状态，并且状态之间的转变依赖于事件的触发。另一方面，原先存在网页上的信息由于事件触发后可能引起页面的 DOM发生变化导致部分信息的丢失，而这些信息可能还没有访问过，会导致获取的资源不完整，因此需要对原先的页面状态信息进行保存，以便于能恢复到之前的节点来访问丢失的资源。

2 基于 Ajax的爬虫关键技术的实现

2.1 嵌入式解析引擎的实现

要想设计一个适合 Ajax的爬虫，首先要解决的问题就是能够对 JavaScript解析执行。虽然普通的浏览器能很好地解决脚本解析问题，但是存在着爬虫无用的功能，例如：页面的图像渲染，元素的位置排版，多媒体资源（图像文件，flash等）的下载。因此，本文设计了一个嵌入式引擎作为爬虫的核心组件，是驱使爬虫工作的重要部分，主要功能有支持将页面解析成 DOM树，支持 DOM的动态操作，能够对 JavaScript脚本的解释执行，XMLHttpRequest对象的方法，支持事件触发等，设计中参考了嵌入式浏览器一些设计思想[9]。嵌入式引擎的体系结构如图 1所示。

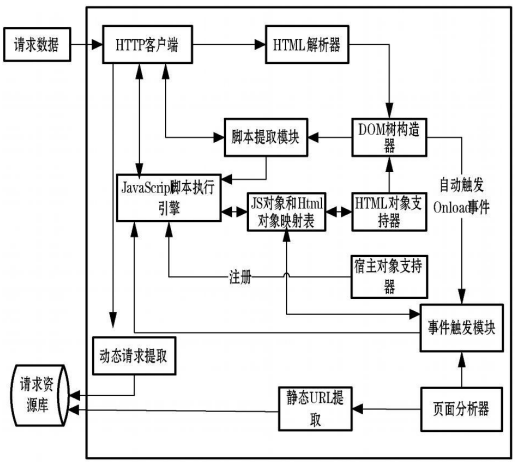

图 1　嵌入式解析引擎的体系结构

HTTP客户端　负责向服务器发送请求（包括：页面请求，Ajax异步请求，对需要的 JS文件请求），接受服务器发来的数据。

HTML解析器　将 HTTP客户端接受页面，将其解析成规范的 XML文档，方便建立 DOM树。

DOM构造器　负责将解析完成的页面文档构造成 DOM树和由于脚本引起的 DOM树的部分更新，同时也负责将分析到的每一个 html元素利用 html对象支持器构建 html对象，并将所有的构建好的 html对象组织成树型结构，确立每一个元素的父子兄弟关系。在 DOM树构造完毕后，要触发 onload事件来获取后加载内容，并更新 DOM树。

脚本提取模块　构建 DOM树的同时分析到脚本元素将提取直接或者间接的 JS代码（以 src属性给出文件位置，则通知 Http客户端请求脚本文件），并将 JS代码交给 JavaScript脚本解析引擎去解释执行。

脚本解析引擎　该文采取了 Mozilla提供的 Rhino[8]，该引擎遵循 ECMAScript标准，且允许自定义宿主对象的属性和方法[10]，能够自动地在自定义的 Java对象和 JavaScript对象之间根据名称一致性的原则实现属性和方法的对应，例如当在脚本片断中执行 document.getElementById("a")，那么就会执行 Java中自定义 Document类的 getElementById方法中定义的行为描述，当然需要在 Rhino执行环境的上下文中注册这些对象原型的方法和属性。

Html对象支持器　定义了每一个类型的 html元素的类，在类实现中包含了对 DOM树进行实际操作的方法，包括查找、修改、删除、添加 DOM树中的元素，例如代码 document.getElementById("id").innerText=text;先执行宿主对象 Document的 jsFunction_getElementById方法，而在 getElementById方法实现中先找到对应的 Document的 html对象，执行该 html对象实现的根据 ID属性查找元素的方法，从 DOM树中找出对应的 html元素，并返回对应的 JS对象，执行该对象的 jsSet_innerText，而在 jsSet_innerText方法实现中又需要利用对应的 html对象的方法来动态地修改 DOM树，因此该模块成为 JavaScript解析引擎动态操作 DOM树的纽带。

宿主对象支持器　实现在脚本解析中需要用到的宿主对象的属性和方法。由于部分脚本的方法和属性在不同的浏览器中存在着差异，考虑到 IE是使用最广泛的浏览器，在具体实现中遵循 IE浏览器的标准，主要宿主对象有：（1）DOM对象，实现 W3C制定的 DOM Level1（包括 core DOM和 html DOM）提供的接口；（2）BOM对象（浏览器对象模型）；（3）Event对象，目前支持 MouseEvent（例如 click，mouseover等）和 HTMLEvent（例如 load，unload，change等）；（4）CSS对象，考虑到脚本中特别是 Ajax框架中使用了对元素的样式操作，但是对于爬虫来讲，毫无意义，因此进行简化处理，只定义方法和属性，但是不实现其功能，方法体设置为空，读属性操作返回默认值；（5）XMLHttpRequest对象，模拟 IE的 ActiveXObject对象来实现，并自在执行 XMLHttpRequest对象的 send方法时，需要通知 http客户端进行 HTTP请求。

页面分析器　爬虫可以从构建好的 DOM树提取出可触发事件的元素，也可以提取出页面存在的超链接和表单的 URL（以备传统的爬虫使用）。

事件触发模块　页面分析器利用元素的 XPath找到可触发事件 html元素，利用映射表对应的 JS对象，根据事件类型构建事件对象，用 html元素的事件属性值构造该对象的事件处理函数，在执行引擎中执行该 JS对象下的要触发事件的对应的处理函数。

请求资源库　数据源主要来自两个途径，一个来自动态地从 Http客户端发送的请求中捕获，另一个来自页面分析器提取的 URL，资源库设计了如下字段信息来存放：获取时间，请求方式，URL，请求体。

2.2 状态信息的保存

通过对 Ajax应用程序的分析，可以将每一次事件触发更新后的页面所对应的 DOM树看作是一个状态节点，一个事件的触发就是引起一个状态转移的条件，成为状态图的一条边，整个 Ajax网站就构成了一个有向状态图，可以用一个二元组＜V，E

?1994-2015 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

>来描述，其中，V代表状态节点的集合，集合中的每一个元素为一个状态节点；E代表连接两个状态节点的有向边的集合，而集合中的任意元素 v(s1, s2) V, s1 S, s2 S则对应一个使 s1转换到 s2的可触发事件的元素 ，那么一条有向边就对应页面中的一个可触发事件的元素。对于状态信息的保存问题就转化成对状态图的保存，而对图的保存就是点和边的保存，从而又可以转化成对于每一个状态节点，每一条边的信息保存。在具体实现中，一个状态节点的信息包括：节点 ID(状态名称)和对应页面的文档表示，一条有向边的信息包括：触发事件的元素名称、属性、在页面的 XPath定位表示(以便找到元素)、对应的事件类型(例如 onclick)、起点的 ID和终点的 ID。当需要进行状态节点恢复时，首先从生成的有向图中找到一条从起始节点到要恢复的状态节点的路径，爬虫利用嵌入式引擎加载初始 URL开始，对路径上的每条边的对应的事件进行回放，实现到达要恢复的节点的状态。

3 爬虫算法描述

结合上面阐述的关键技术，提出了适合 Ajax的爬虫模型的算法流程，分为两大过程：爬虫初始化和爬虫工作主流程。

3.1 爬虫初始化

在爬虫工作开始之前，需要进行初始化工作，包括初始化一个嵌入式解析引擎(一些引擎工作前所需的准备工作，如在脚本执行环境下注册定义的宿主对象原型、加载映射表等)，初始化一个有向状态图，建立一个全局事件元素集合。该集合记录着爬虫已经访问到的所有的可触发事件的元素信息(包括元素名称、属性、在页面的 XPath表示、对应的事件类型)，并随着爬虫的爬行进行增量更新；然后加载需要爬行的 Ajax网站的初始 URL，根据构造的对应的文档构建有向状态图的起始节点，并设置为当前作用节点，至此爬行初始化工作结束。

3.2 爬虫工作流程

初始化工作完毕，爬虫工作正式开始，流程如图 2所示。

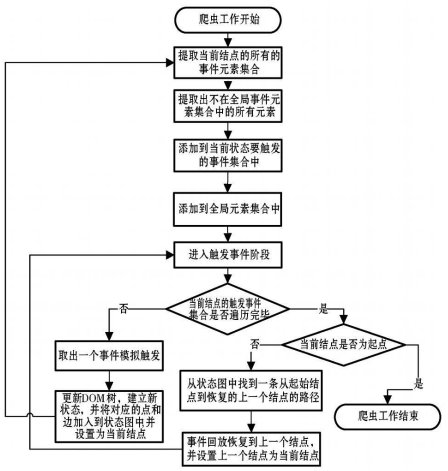

图 2 爬行算法描述

4 对比实验及实验结果分析

该爬虫模型使用 Eclipse3.2.2开发工具开发。为了测试该模型的有效性，以一个采用了 Ajax技术的 blog网站——天枫 blog beta1.0作为实验对象，将其搭建在 IIS V5.1上进行测试。该文设计了一组对比实验，采取三种不同的爬虫算法，第一种是传统的爬虫算法。第二种是使用了嵌入式引擎解析脚本，但是没有采用图 2的算法，而是采用如下过程：(1)提取出当前的页面所有的还没有发现的可触发事件元素，按照找到的顺序存入队列；(2)从队头取出一个元素，如果该元素存在于当前页面，则触发对应的事件；若不存在，再从队列弹取一个元素，直至获取一个当前页面存在的元素，触发之；重复(1)、(2)直到队列为空。第三种是采用了图 2的算法。实验结果如表 1所示，为了表述简单，将上面描述的三种爬虫分别称为爬虫Ⅰ，爬虫Ⅱ，爬虫Ⅲ。

表 1 对比实验分析

爬虫方式	静态 URL(只包含页面 URL)	动态请求(不重复)		状态节点数	有向边数
		Get	Post		
爬虫Ⅰ	3	0	0	不存在	不存在
爬虫Ⅱ	3	12	1	不存在	不存在
爬虫Ⅲ	3	28	8	78	77

从表 1的实验数据中发现，传统的爬虫只能够静态地分析网页中的资源信息(只包括超链接)，但是在完全使用了 Ajax技术的网站中，无法提取脚本资源，主要原因是无法解析隐藏在脚本中的异步请求。对于爬虫Ⅱ支持了脚本的解析，能够提取隐藏在脚本中的异步请求的 URL，但是由于没有实现对前一个结点信息的恢复功能，导致了一些状态结点的信息不能完全提取，正如表 1所示爬虫Ⅱ的状态提取的请求资源少于使用了状态信息保存的爬虫Ⅲ提取的数量。通过这三组对比实验，说明要完整地爬行 Ajax网站，必须解决脚本解析和状态信息的保存和恢复两大问题，同时也说明本文提出的爬虫模型是行之有效的。本文设计的爬虫，不仅提取出了静态的 URL，也提取出了脚本动态交互的请求信息，同时保存的状态图表示了 Ajax网站的体系结构，以特定事件取代了以往以静态 URL作为状态页面之间联系的桥梁，还可以利用状态图的状态结点中的 DOM信息抽取出脚本隐藏的 Web内容。另外，需要指出的是状态节点数目大于 Ajax异步请求的数量主要是一些事件触发没有引起请求，而是页面本地化修改，例如设置 CSS的特性、列表的隐藏与显示等。

5 结论与展望

针对目前爬虫利用 URL来爬行的工作机制不能有效地对 Ajax网站进行爬行，本文提出了一个基于事件驱动的爬虫模型，能够较为有效地爬行 Ajax网站。同时该模型的设计在脚本解析的正确完整性以及算法的效率上还有待进一步完善。另外，目前的爬虫只针对完全使用 Ajax技术的网站，设计出一个对于 Web1.0和 Web2.0都适用的爬虫模型也是下一步工作的重点。

参 考 文 献

[1] Raghavan S, Garcia-Molina H. Crawling the hidden web[C]//Roma, Italy: Proc of the 27th International Conference on Very Large Data Bases(VLDB), 2001: 129-139.

?1994-2015 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

[2] Barbosa L, Freire J. An adaptive crawler for locating hidden-web entry points[C]//Alberta, Canada: Proc of the 16th international conference on World Wide Web, 2007: 441-450.
[3] Ntoulas A, Zerfos P, Cho J. Downloading textual hidden web content through keyword queries[C]//North California, USA: Proc of the 5th ACM/IEEE-CS joint conference on Digital libraries, 2005: 100-109.
[4] Alvarez M, Raposo J, Pan A, et al. Crawling the Content Hidden Behind Web Forms[J]. Lecture Notes in Computer Science, 2007, 4702: 322-333.
[5] Alvarez M, Pan A, Raposo J, et al. Crawling Web Pages with Support for Client-Side Dynamism[C]//HongKong, China: Proc of the 7th International Conference on Web Age Information Management (WAIM06), 2006: 252-262.
[6] 王映，于满泉，李盛韬，等. JavaScript引擎在动态网页采集技术中的应用[J]. 计算机应用，2004, 24(2): 33-36.
[7] 罗兵. 支持 Ajax的互联网搜索引擎爬虫设计与实现[D]. 杭州：浙江大学，2007.
[8] Mozilla Tutorial. Embedding Rhino[EB/OL]. 2006-11-14. http://www.mozilla.org/rhino/tutorial.html.
[9] 李英各. 嵌入式浏览器中 JavaScript的设计与实现[D]. 成都：电子科技大学，2004.
[10] Mozilla Rhino documentation[EB/OL]. 2008-4-14. http://developer.mozilla.org/en/docs/Rhino_documentation.

（上接第 39页）

$$\tau_{ij}(t+1) = \rho\tau_{ij}(t) + \Delta\tau_{ij} \tag{3}$$

其中 ρ为信息素挥发系数，$\Delta\tau_{ij} = \sum_{k=1}^{m}\Delta\tau_{ij}^{k}$，$\Delta\tau_{ij}^{k}$为第 k只蚂蚁信息素的更新：

为使可行解对应边上信息素加强，劣质解对应边上信息素削弱，本文采用。

$$\Delta\tau_{ij}^{k} = \begin{cases} Q \cdot R(k) & \text{如果蚂蚁 } k\text{经过}(i,j) \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

Q为一正数，R(k)为第 k只蚂蚁搜索到的子系统可靠度。

3.4 算法步骤

Step1 初始化。输入蚂蚁数 n，Q，M，α、β、γ、ρ，时间 t=0，循环次数 $N_c=0$，设置最大循环次数为 N_{max}，子系统最大冗余数 n_{max}，$\tau(0)=0$，$\Delta\tau_{ij}(0)=0$。

Step2 根据 n和 n_{max}生成集合 V_i，并计算 C(i)和 R(i)。

Step3 将 n只蚂蚁均匀放在 n个子系统上。

Step4 对所有蚂蚁根据式(1)、(2)计算转移概率，根据式(3)、(4)更新所选路径上的信息素，构造问题的解。

Step5 计算目标函数值 C_s，如果 $C_s(t+1) < C_s(t)$，更新目标函数值，更新问题的解，否则重复 Step4。

Step6 如果已经达到最大循环次数，循环结束，输出目标函数值和问题的解，否则回 Step4。

4 工程应用与实例分析

本文用蚁群算法解决文献[3]提出的问题（不考虑重量），系统参数如表 1所示。取 n=3，Q=10，M=100，ρ=0.85，N_{max}=300，n_{max}=5，R_0=0.98。用 Matlab编程实现算法，在第 73次循环第一次得到最优解（2，4，4|1，3|5，5，5，5，5），最优系统可靠度为 0.9816，总费用为 30，最优解率接近 70%，平均解为 0.98149。最佳方案图如图 3所示。

表 1　实例参数系统

	子系统 1			子系统 2			子系统 3		
	R	C	W	R	C	W	R	C	W
1	0.94	7	4	0.97	6	5	0.96	7	6
2	0.91	6	6	0.86	3	7	0.89	6	8
3	0.89	6	8	0.70	2	4	0.72	4	5
4	0.75	3	7	0.66	2	7	0.71	3	4
5	0.72	3	8	0.65	2	6	0.67	2	4

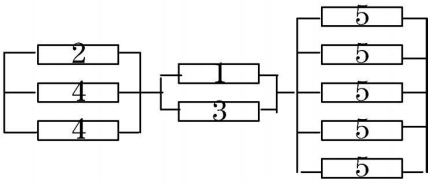

图 3　实例最佳方案图

如果取 n_{max}=3，R_0=0.97，则在第 41次循环第一次得到最优解（3，4，4|2，2，4|1，4），最优系统可靠度为 0.9751，总费用为 30，平均解为 0.9687。最优解率达到 86%，由此可见，降低系统可靠度和系统复杂度的要求，可使蚂蚁在相对较短的时间内找到问题的最优解。

5 结束语

本文将蚁群算法应用串并系统的可靠性冗余分配模型中，成功解决了串并系统冗余优化问题。蚁群算法与其它元启发式算法一样，可以有效地克服求解组合优化的计算复杂度问题。蚁群算法是一种新兴的群体智能算法，许多研究已经证明，蚁群算法具有很强的发现较好解的能力。蚁群算法凭借其优异的算法性能和算法特点很快成为启发式方法范畴内的一个独立分支，在有关国际会议上多次作为专题加以讨论。自 1998年第一次国际蚁群算法国际会议召开后，蚁群算法更是成为智能仿生算法的研究热点，越来越多的研究人员正在从事这方面的工作。

参 考 文 献

[1] Dorigo M, Gambardella L M. Ant colony system: a cooperative learning approach to the traveling salesman problem[J]. IEEE Trans on Evolutionary Computation, 1997, 1(1): 53-66.
[2] 高尚，杨静宇. 群智能算法及其应用. 中国水利水电出版社，2006.
[3] Chern M S, AE Smish. Stochastic Formulations of the redundancy allocation problem[C]//Proceeding of the fifth industrial Engineering Research Confreence (IERC) MINneMNMay, 1996: 459-463.
[4] 王正初，李薇薇. 基于粒子群的可靠性优化[J]. 台洲学院学报，2006, 28(6): 29-32.
[5] Dorigo M, Gambardella L M, Thomas Stützle. 蚁群优化[M]. 张军，等译. 清华大学出版社，北京，2007.
[6] 赵建民. 复杂系统可靠性冗余的平衡优化方法[J]. 机械强度，2000, 22(4): 283-285.
[7] 黎放，朱承，王威. 冗余系统费用模型及优化[J]. 系统工程及电子技术，2008, 22(9): 84-89.
[8] 刘立东，蔡淮. 基于遗传算法的混合蚁群算法[J]. 计算机工程与设计，2008, 29(5): 1248-1250.

?1994-2015 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net