第 32 卷 第 19 期 计 算 机 工 程 2006 年 10 月
Vol.32 № 19 Computer Engineering October 2006
·软件技术与数据库· 文章编号：1000—3428(2006)19—0049—02 文献标识码：A 中图分类号：TP311

OnceSAXParser：一种高效轻量的 XML 解析器

曹冬磊，田四化，金蓓弘
(中国科学院软件研究所软件工程技术中心，北京 100080)

摘 要：通过优化 XML 词法和语法处理以及构造轻量级体系结构，实现了支持 SAX 的高效 XML 解析器——OnceSAXParser。文中还将 OnceSAXParser 与目前最流行的 XML 解析器 Xerces 进行了对比分析和性能测试，结果显示 OnceSAXParser 的性能比 Xerces 平均提高了 27%以上。

关键词：XML 解析器；SAX；性能优化

OnceSAXParser: An Efficient and Light-weighted XML Parser

CAO Donglei, TIAN Sihua, JIN Beihong
(Technology Center of Software Engineering, Institute of Software, Chinese Academy of Sciences, Beijing 100080)

【Abstract】 Xerces is the most popular and best performed parser till now. But its architecture is so complicated that it limits its performance. By optimizing lexical and syntax processing for XML, along with constructing the light-weighted architecture, an efficient XML parser—OnceSAXParser is implemented. The paper compares it with Xerces which is a most popular XML parser from the view of structure analysis and performance test. The results of performance tests are given to show that OnceSAXParser is 27% faster than Xerces on the average.

【Key words】 XML Parser; SAX; Performance tuning

XML(eXtensible Markup Language)是 W3C(World Wide Web Consortium)组织定义的一种数据传输和交换的标准语言，它定义严格，结构清晰，并且采用纯文本方式保存。

W3C分别于 1998 年和 2004 年推出了XML1.0 和 1.1 规范。为了测试XML的兼容性，W3C提供了XML兼容性测试软件包XMLConf[1]。其中包含了来自多个组织、公司或个人的 2 000 多个测试用例，为XML解析器的开发者提供了相对完备的XML兼容性测试准则。

目前比较流行的XML解析器是Xerces[2]，它是Apache组织提供的具有世界一流水平的用于解析和创建XML文档的软件包，提供了XML文档的验证功能，具有很好的模块化以及可配置性，并且提供了对XMLSchema的支持。目前最新的版本是Xerces2 Java Parser 2.6.2。经过测试发现，Xerces能够通过XML兼容性测试中的绝大部分测试用例，而且性能优异。

1 XML 解析器设计

1.1 XML 语言分析

XML规范采用EBNF(Extended Backus Naur Form)表示语言的产生式，一个良构的XML文档就是由符合这些产生式和若干良构性约束的数据组成。XML采用EBNF表示，所以这些产生式所表示的文法等价于上下文无关文法，可通过构造一个下推自动机来识别XML文法[4]。XML产生式可分成 2 类：一类以小写字母开头的单词作为产生式的左部，如：

elementdecl ::= '<!ELEMENT ' S Name S contentspec S? '>' (1)

另一类是以大写字母开头的单词为产生式的左部，如：

Name ::= (Letter | '_' | ':') NameChar* (2)

按照 XML 规范的约定，这类产生式是正则表达式，可以通过一个确定的有限自动机来识别。XML 规范中的产生式表示了 XML 文档的逻辑结构，对 XML 文档的解析，就是对各种产生式的识别。通过对产生式的识别，可以获得 XML 标记(markup)和字符数据(character data)。通过分析，发现 XML 标记的限界符都是确定的。

1.2 词法分析器设计

词法分析的任务是读输入字符流，产生用于语法解析的记号(token)序列。通常将词法分析器作为语法解析器的一个子程序来实现。由于 XML 语言中所有标记的限界符都是确定的，因此可以把限界符的匹配工作直接交给语法解析器去完成。因为 XML 规范中仅定义了 12 种标记，所以，语法解析器对这 12 中标记的判断的开销并不是很大。而由语法解析器主动识别标记的方法不但可以简化词法分析器的设计，而且可以减少语法解析器对词法分析器的调用开销。这是因为在 XML 文档中各种标记的重复度很高。例如，一个典型的大小为 100kB 的 XML 文档中含有 20 000 个标记，而语法解析器每次主动识别到一个标记，可以节省一次调用词法分析器去识别的开销。

1.3 语法解析器设计

语法解析器分成 DTDParser 和 SAXParser 两部分，分别解析文档的 DTD 部分和非 DTD 部分，它们均是通过构造下推自动机来实现语法解析的。图 1 给出了解析产生式(1)的自动机的状态转换图。因为设计不要求进行有效性(Validity)验证，所以 DTDParser 只需要记录实体声明和元素列表声明中的信息。比对 Xerces，它也实现了一个专门解析 DTD 部分的解析器，但是由于需要做有效性验证，因此它需要记录 DTD 中的所有标记声明的信息(包括实体声明、属性列表声明等)，

基金项目：国家“973”计划基金资助项目(2002CB312005)；国家“863”计划基金资助项目(2001AA113010)
作者简介：曹冬磊(1980－)，男，硕士生，主研方向：分布式计算；田四化，硕士生；金蓓弘，博士、副研究员
收稿日期：2005-10-19 **E-mail：**caodl@otcaix.iscas.ac.cn

保存的数据较多。

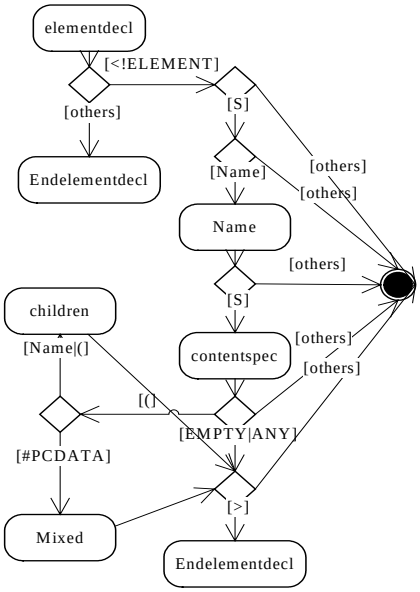

图 1 解析 elementdecl 的状态转换

2 XML 解析器实现

本文实现了一个 Java 版本的支持 SAX 接口的 XML 解析器——OnceSAXParser。SAX 接口是一种基于事件的解析方式，通常用于对性能要求较高或硬件资源有限的系统中。根据上面的讨论，采用基于 XML 文档逻辑结构的解析方式，逐条解析 XML 文档中的标记。图 2 给出了 OnceSAXParser 的主要类及其类之间的关系，其中 SAXParserImpl 用于实现 JAXP(Java API for XML Processing)接口定义的 javax.xml.parsers.SAXParser，SAXParser 用于实现 SAX2.0 接口定义的 org.xml.sax.XMLReader，Scanner 用于词法分析，EntityInfo 用于保存实体的信息，这些类以紧耦合方式工作。

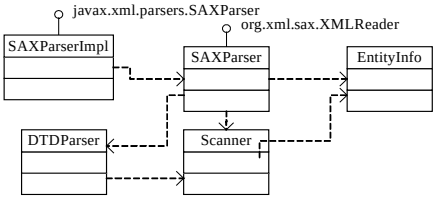

图 2 OnceSAXParser 的类结构关系

与 OnceSAXParser 的实现不同，Xerces 定义了一套内部接口 XNI(Xerces Native Interface)，Xerces 的语法解析器解析 XML 文档产生的结果被封装成符合 XNI 定义的事件。在 XNI 与 SAX 接口之间，Xerces 实现了一个 SAXParser 类，用于将 XNI 提供的事件转化为符合 SAX 接口的事件。XNI 可以增加系统的灵活性，例如 Xerces 在 XNI 的基础上实现了一个 DOMParser 类，它可以根据 XNI 提供的事件信息，构造出一个符合 DOM 规范的 XML 解析器。但是，通过一个内部接口来增加系统的灵活性会造成性能上的损失，不利于支持 SAX 的应用。OnceSAXParser 选择了紧耦合的结构能有效避免此类问题。

在接口的实现上两者也采用了不同的方法。J2SE 的 org.xml.sax 包中定义了与 SAX 解析相关的一系列接口和类，其中在 SAX1.0 中，所有的 SAX 解析器都遵守 org.xml.sax.DocumentHandler 接口定义的回调函数，在 SAX 2.0 中该接口被 org.xml.sax.ContentHandler 替代。目前，基于 SAX 的大部分应用都使用 SAX 2.0 接口，所以 OnceSAXParser 直接实现了 ContentHandler 接口，同时实现了一个 XMLReader Adapter 类，该类利用适配器模式，将 ContentHandler 接口定义的事件适配为 DocumentHandler 接口定义的事件。而 Xerces 是用一个 AbstractSAXParser 类封装了 ContentHandler 接口和 DocumentHandler 接口定义的回调函数。这样，在解析过程中每产生一个事件，都会同时产生符合 ContentHandler 和 DocumentHandler 接口的事件。虽然这种方式同样能够保证对 SAX 1.0 的兼容，但是绝大多数情况下，这种方式会影响解析的性能。

3 性能优化

为了实现对 XML 文档的高效解析，采取了一些优化措施，除了前面讨论过的设计上的优化外，另外还有下面给出的一些实现上的重要优化措施。

3.1 高效的解码器

按照 XML 规范，解析器要能够正确处理 UTF-8 和 UTF-16 两种编码的 XML 文档。UTF-16 是一种双字节的定长的编码方式，其解码比较简单。而 UTF-8 是一种变长的编码方式，对其解码有一定的复杂性。经过测试，Java 语言提供的 UTF-8 解码器的效率无法满足要求，所以，作者实现了一个专门用于 UTF-8 编码的解码器——UTF8Reader。表 1 给出了使用 UTF8Reader 前后读取一个约 1MB 的 XML 文档 1 000 次所花的时间，可以看到，使用 UTF8Reader 可提高读取性能。

表 1 使用 UTF8Reader 前后的性能比较

测试	不使用 UTF8Reader	使用 UTF8Reader	性能提升
1	53 781ms	41 344ms	23.13%
2	53 765ms	41 297ms	23.19%
3	53 843ms	41 531ms	22.87%

3.2 浓缩处理越界判断

在解析一个 XML 文档时，不可能每次都将整个文档全部装入内存，所以必须设置一个有限的缓冲区，并将 XML 文档分批装入。这使得词法分析器每读取 1 个字符时，都要判断当前读指针是否到达缓冲区的结尾，即是否越界。根据统计，词法分析过程中大部分的开销都用于对匹配产生式(2)的字符串的判断，而这其中又有一部分开销用于读指针是否越界的判断。为此实现了一个算法来减少这种开销，首先假设绝大部分的名字字符串的长度小于 20 个字符，算法如下：

(1)如果缓冲区的剩余空间足够容纳 20 个字符，跳至步骤(3)；

(2)保留还未解析的内容(少于 20 个字符)，刷新并填充缓冲区；

(3)在 20 个字符范围内，直接进行对匹配产生式(2)的字符串的判断。如果这 20 个字符之内能够成功匹配，则结束；否则，读指针后移 20 位，并保存已经匹配的名字字符串，同时跳到步骤(1)。

通过这个算法，我们实现了对频繁执行的判断操作的“浓缩处理”，即，将每读取一个字符判断一次读指针越界操作，变为每读取 20 个字符判断一次指针越界操作，这种方式可有效减小开销。

3.3 基于统计规律的标记识别优化

由于语法解析器是针对 XML 文档中的标记进行逐条解析的，因此对各种标记的识别效率将直接影响到解析器的效率。由于不同的标记在 XML 文档中出现的频率差异很大，因此，在 OnceSAXParser 的实现上对这一部分做了优化，对一些典型的 XML 文档进行统计，发现一个 XML 文档中开始标签(start-tag)、结束标签(end-tag)和字符数据(character data)出现的频率最高，所以在语法解析流程中将这些标记的识别放在优先的位置。表 2 给出了各种标记出现次数的统计结果。

（下转第 53 页）