大型 XML 文档解析技术的应用与研究

张太彪[1], 曾文华[1*], 陈志伟[2]
(1. 厦门大学 软件学院, 智能信息技术福建省重点实验室, 2. 厦门大学物理与机电工程学院, 福建 厦门 361005)

摘要: 随着 XML 应用的不断深入, XML 文档快速解析技术的研究成为了当前的热点问题. 在研究 XML 相关解析技术的基础之上, 讨论大型 XML 文档的解析和处理问题, 寻求实际问题的最优解决方案. 首先, 对目前较适合大型 XML 文档解析的两种技术(VTD-XML 和 SAX)做了简要的介绍, 对其各自的优劣性进行了比较和分析; 其次, 针对作者正在研发的"核磁共振谱仪控制软件系统"中参数的 XML 设计和处理问题提出了具体的解决方案, 验证了它们在处理大型文件时的可行性, 并对它们的实际效果进行了对比分析; 最后, 针对以上两种解决方案的不足之处, 提出了大型 XML 数据文件处理的其他解决方案.

关键词: 大型 XML 文档; XML 解析器; SAX; VTD-XML
中图分类号: TP 312 文献标识码: A 文章编号: 0438-0479(2009)03-0338-04

随着信息技术的不断发展, 可扩展标记语言(Extensible markup language, XML)作为集数据表示、存储、传输和处理为一体的工具, 在信息世界的各领域发挥着越来越重要的作用. 不过, 由于 XML 本身只是一种以纯文本对数据进行编码的格式, 要想利用其中所编码的数据, 必须借助解析器将这些数据从纯文本中解析出来. 因此, 如何高效地解析 XML 文档至关重要.

在此之前, 许多人对 XML 的解析技术都很关注, 不少文章都探讨了 XML 快速解析的有关问题, 但对于大型 XML 文档解析的讨论不多(这里的大型 XML 文档是指几十兆至几百兆的 XML 文档). 在日常的 XML 应用中, 有时会碰到需要处理大型 XML 文档, 或考虑到将来可能会用到大型 XML 文档的情况, 这时必须选取能满足大型文档解析需求的技术. 当前, 各种 XML 解析技术都有自己的适用领域, 但并非所有技术都适合大型 XML 文档的解析. 事实上, 目前真正能适合大型 XML 文档解析的技术只有 XML 简单应用程序接口(Simple API for XML, SAX)[1] 和 XML 虚拟令牌描述(Virtual token description for extensible markup language, VTD-XML)[2] 两种, 因为文档对象模型(Document object model, DOM)解析技术在解析时必须在内存中一次性构建文档对象树, 然后通过树来访问具体的元素, 其内存的消耗量一般是原文档大小的 4~10 倍, 对于大型的 XML 文档无法正常解析.

1 目前较适合大型 XML 文档的解析技术及其性能比较

1.1 SAX

和大家所熟知的 DOM 不同, SAX 是一种基于事件模型的解析技术, 它虽然不是万维网协会(World wide web consortium, W3C)提出的官方标准, 但却因为支持部分解析和内存消耗较小等优势, 在实际中被广泛应用, 几乎所有的解析器都对其提供支持.

SAX 提供了一种顺序访问 XML 文档的方式, 整个 XML 解析过程有点类似于流媒体的处理过程. 在解析时, XML 文档被从头到尾顺序地读入, 触发一系列相关的事件, 同时发出相应的 SAX 事件报告, 应用程序只需根据事件的发生来调用相应的处理方法, 对每一事件进行处理(当然, 前提是程序员必须在此过程中对自己感兴趣的事件进行相应的编码).

一般来说, SAX 按照下列步骤来解析 XML 文档[1]: (1) 设置事件的处理器, 即对 ContentHandler 接口做相应的实现; (2) 产生解析器, 载入需要解析的文档并注册相应的事件处理器; (3) 在感兴趣的事件方法中加入适合程序需要的控制逻辑, 如果需要的话, 可以产生自己的对象模型; (4) 根据文档解析过程中产生的事件, 调用相应的事件处理方法; (5) 重复(4) 直至文档解析结束.

从以上分析不难看出, 在使用 SAX 解析时, 应用程序只是在读取数据时检查数据, 不必将数据存储在内存中(这一点基于树形结构的解析技术是没办法做

收稿日期: 2008-09-28
基金项目: 国家科技支撑计划(2006BAK03A22)资助
* 通讯作者: whzeng@xmu.edu.cn

到的); 另外, 可以由开发人员自己来决定所要处理的标记, 并在某些条件得到满足时停止解析, 因此, 用它可以解析大于系统内存的 XML 文档, 这些对于大型文档来说都是巨大的优点.

1.2 VTD-XML

传统的 XML 解析器, 无论是 SAX 还是 DOM, 均是基于提取解析(Extractive parsing) 模式. 这种解析模式的特点是在解析 XML 文档时, 提取一部分原文件, 一般是一个字符串, 然后在内存中对其进行对象的构建, 解析的效率相对低下. 为了摆脱提取式解析带来的瓶颈, 一种与之不同的解析模式——非提取式解析(Non-extractive parsing) 模式随即诞生. 这种解析模式在解析 XML 时, 将文档作为一个整体一次性读入, 以二进制数组的形式来处理 XML 数据, 避免了大量对象的创建, 解析效率大大提高, 为大型 XML 文档的解析提供了必要的条件.

VTD-XML 是近年兴起的一种新型的 XML 快速解析技术, 它基于非提取的 XML 解析模式[3]. 作为一种以文档为中心的 XML 解析器, 它克服了 DOM 和 SAX 的一些问题, 通过数组的方式在内存中实现了 XML 的快速检查, 并对 XPath 查询提供了支持.

与 SAX 不同的是, VTD 并不是一个 API 规范, 它仅仅是关于如何编码令牌中各种参数的二进制格式说明. 一个 VTD 记录采用原始的数值类型作为记录类型, 其中包含了某一元素的起始位置、长度、令牌类型和其在 XML 中的深度等信息, 具体比特层格式如图 1 所示[2]. 对于某一特定的类型, 进一步将其长度划分成前缀长度和序列名称长度, 它们共用一个起始位置. VTD 记录通常按文档中的顺序原封不动地在内存中存放各元素, 其中顶层 VTD 保存有整个源文档的完整信息.

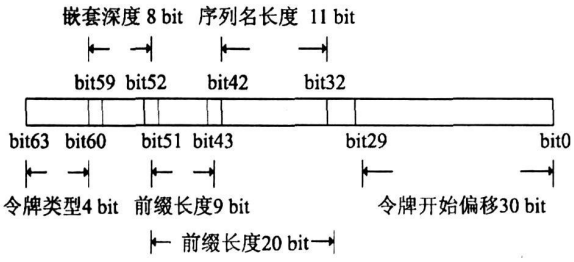

图 1　VTD 记录的比特层格式

Fig. 1　Bit-layer format for VTD record

VTD-XML 采用 VTD 来记录 XML 解析所需的信息, 克服了传统解析器过多创建对象的问题. 另外, 由于 VTD 的长度是固定的, 所以读取和查询的效率较高. VTD-XML 通过对 VTD 记录进行遍历, 来找到所需的 XML 元素. LC 机制将 VTD 以其深度作为标准构建的一个树形的表结构. 只需利用相关的信息, 便可以在最少的几步内查找到需要的元素, 遍历的性能十分突出.

VTD-XML 作为目前世界上最快的 XML 解析器[4], 采用的是一种全新的解析思想, 无论在处理时间方面, 还是在内存使用方面均具有相当高的效率, 适合大型 XML 文档解析. 另外, VTD-XML 在解析和处理 XML 语义方面比 DOM 和 SAX 弱得多, 因此是一种很有潜力的 XML 处理技术.

1.3 SAX 和 VTD-XML 的解析性能比较

关于这两种解析技术的性能比较, xsimpleware 的官方发布了相应的对比数据[3], 具体如表 1 所示(这里的 SAX 速度是指 SAX 解析中没有插入任何额外的处理逻辑, 也就是 SAX 的最高速度. 所以, VTD-XML 的速度性能在实际中还要更优于 SAX).

表 1　VTD-XML 和 SAX 的性能比较

Tab. 1　Performance comparison between VTD-XML and SAX

性能指标	VTD-XML	SAX
解析速度	1.3~ 1.5 倍于 SAX	快, 线性速度
内存使用	1.5~ 2 倍于文档	不随文档变化
易用性和可维护程度	简单, 易用维护	很差
随机访问	支持	不支持
增量更新	支持	不支持
避免每次都解析文档	支持	不支持
硬件支持	支持	不支持

2 核磁共振谱仪软件中参数的 XML 设计和处理方案实现

2.1 项目中的 XML 参数文件的设计和处理

核磁共振(Nuclear magnetic resonance, NMR) 谱仪软件是一个完成谱仪控制、实验设计和实验数据处理与显示等诸多功能的系统, 属于谱仪系统的软件分支. 该系统中, 从仪器的控制到实验的设计, 再到数据处理、显示都离不开参数, 因此参数设计成为这一软件系统中的一个核心环节. 目前流行的核磁共振控制软件(瓦里昂和布鲁克两家公司针对各自生产的仪器都有自己的控制软件) 参数设计均使用普通的文本文件, 这种设计方式的主要缺陷是无法描述复杂结构的参数文件, 不能满足用户提出的复杂功能需求. 相比而言, XML 作为与平台、语言和协议无关的格式描述和数据交换的标记语言, 可以满足系统参数设计的复杂度和

跨平台等多方面需求,因此将它用于系统的参数设计模块中.

为了使系统可以正确并且高效地运行,选取合适的 XML 解析技术成为系统的关键点之一. 在本系统的实际运行中,用户每一次实验均可能对数据进行多次采样,因此 NMR 的自由感应衰减(Free induction decay, FID)数据对应会有多个 block(段),为了完成对不同 block 中 fid 数据的处理和显示,在参数文件中必须为每个block 维护各自的参数,当 block 的数量达到一千个甚至几千个时,XML 参数文件的大小也因此变得很大,可达到了几十兆甚至上百兆. 系统设计之初,出于方便采用了 DOM 作为 XML 参数文件的解析方案,对于简单的实验系统能正常运行,可是对于稍大的文档系统运行速度开始变慢,当文件大小达到 10 MB 后,系统会因内存不足而无法运行. 显然,这不能满足系统的需求,采用适合大型 XML 文档的解析技术势在必行.

经过不断的对比和研究,系统决定选取 SAX 或 VTD-XML 为 XML 的解析方案,下面将对比这两种实现方案的具体使用情况.

2.2 两种技术方法的实际效果对比及分析

表 2 是通过对两种设计方案的实验测试所得出的结果,其中层数代表某个实验的规模大小(单位为层),文档大小代表的是 XML 参数文件的大小(单位为 kB),耗时代表解析文档所花的时间(单位为 ms),耗存代表解析文档所消耗的内存大小(单位为 MB).

由得出的结果不难发现,在实际应用过程中,VTD 由于无需产生额外的对象模型,在时空性能上均优于 SAX. 不过,这两种方案在文件大小达到 100 多兆时出现了内存溢出的情况. SAX 方案发生内存溢出的主要原因在于当文件规模达到一定的大小时,系统无法为所需参数构造完整的对象模型;VTD 方案出现系统内存溢出则是因为构建 VTD 记录和位置缓冲 LC 记录及其它相关对象的内存需求无法得到满足.

从本项目目前的实际需求来看,两种方案均是可行的. 当该项目中数据量不断增大,并出现内存溢出的情况时,必须寻求一些其他的解决方案,以实现项目的扩展需要. 接下来,将介绍几种关于大型 XML 数据处理的其他解决方案.

3 大型 XML 数据处理的其他解决方案

前面介绍的是目前常用的 XML 解析技术,主要是采用提取式或非提取式模式直接解析和操作 XML 文档,操作的方式比较直接,但是面对几百兆或更大的文档同样有可能无法正常处理. 下面介绍另外两种大型 XML 数据的处理方案.

3.1 大型 XML 文件的分割和动态加载

对于上百兆的 XML 文档,如果直接采用上述方案进行处理,巨大的内存开销成为了系统处理的一个瓶颈. 一般说来,XML 应用程序成功与否取决于 XML 文档设计的优劣. 而数据粒度的决定是 XML 文档设计中最困难的事情之一,因为粒度过小会影响到存储的效率,粒度过大则会影响 XML 解析和处理的性能. 对于粒度过大的文档为了有效对其进行处理,可以考虑对其进行适当的分割,然后在使用时进行动态加载. 由于 XML 是语法严格的标记语言,因此分割必然受

表 2 SAX 和 VTD-XML 方案的性能对比

Tab. 2 Performance comparison between VTD-XML and SAX scheme

层数	文件大小/kB	SAX		VTD-XML	
		耗存/MB	耗时/ms	耗存/MB	耗时/ms
1	55	172	0.8785	134	0.4122
10	169	234	0.5237	156	0.3254
100	1303	640	3.4699	188	3.0328
500	6348	2375	16.401	390	14.2685
1000	12654	4484	30.711	609	28.5660
2000	25266	10953	62.919	1031	56.9088
3000	37678	17524	100.67	1643	94.4551
4000	50491	24096	138.42	2354	130.8429
5000	63103	33734	198.79	3097	189.6715
6000	75715	43584	278.30	3784	267.8157
8000	100940		内存溢出	5503	291.6453
10000	162164		内存溢出		内存溢出

到相关规则的约束. 分割完后的 XML 文件块无法采用 SAX 或 DOM 直接解析. DOM 需要一个结构完整的文件才能正确解析, 故其无法解析 XML 块; 而 SAX 必须进行相关的改进方可完成文件块的动态加载, 具体原理详见文献[4], 其中提出和阐述了一种基于 SAX 技术的大型文件的分割和动态加载的解决方法, 可以提供一种大型 XML 处理的解决思路.

3.2　采用数据库方式处理大型的 XML 数据

之前介绍的几种 XML 数据处理方式均直接采用文本的形式对数据进行存储, 然后采用有关技术对其进行解析. 与之不同的另一种方式是采用数据库来解决 XML 数据的存储和处理, 目前主要的做法是将 XML 文档以一定的方式映射到关系数据库或面向对象数据库, 把对 XML 的查询转换为对数据库的 SQL 查询, 再将查询和处理的结果转换为 XML 文档的形式呈现给用户. 显然, 数据库可以完成大量数据的处理, 为处理大型的 XML 文档提供了一种解决方案. 值得特别一提的是使用 Native XML Database 来存储和查询 XML 数据的方法[5], 这种方法和其他方法相比具有很多的优点, 尤其对于处理大型文档, 它能很好地提高性能.

4　结束语

通过对 SAX 和 VTD-XML 的研究和分析以及在具体项目当中的使用情况, 可以得出一个结论: 以上两种解析技术确实可以胜任较大 XML 文档的解析任务, 选用哪一种视具体情况而定, 如果遇到更大的文档, 可以考虑通过上面介绍的其他的方式来加以解决.

值得大家注意的是, Google 公司在 2008 年 7 月 7 日发布了其内部使用的开放源代码数据描述语言 Protocol Buffers[6]. 它是一种可用于通讯协议、数据存储等领域的语言无关、平台无关、可扩展的序列化结构数据格式, 一种可伸缩、高效的、自动化的结构化数据序列化机制, 它比较像 XML, 但是更小, 更快, 更简单, 是一种很适合做数据存储或 RPC 数据交换的格式.

由于此项目目前仍在进行当中, 为了不断提高系统性能, 正在不断地寻求更好的解决方案, 并在接下来的工作中尝试上述多种方法的使用, 继续关注 XML 的解析及相关技术, 期待更多类似于 XML 和 Protocol Buffers 的高效数据存储格式的诞生和发展.

参考文献:

[1] 张迪, 朱敏, 张凌立. 基于 SAX 的 XML 解析与应用[J]. 计算机与数字工程, 2008(7): 103-106.

[2] Zhang Jimmy. Simple XML processing with VTD-XML [EB/OL]. [2006-03-27]. http://www.javaworld.com/javaworld/jw-03-2006/jw-0327-simplify.html.

[3] Zhang Jimmy. XML processing for the future [EB/OL]. [2008-06-24]. http://www.codeproject.com/KB/XML/xml_processing_future.aspx.

[4] 孙静, 宋扬, 胡金星, 等. 大型 XML 文件的分割和动态加载研究[J]. 计算机工程与应用, 2003(16): 107-109.

[5] 李鹏飞, 吴洁, 丁秋林. 关于处理大型 XML 数据的 NXD 方法研究[J]. 计算机技术与发展, 2006, 16(3): 179-184.

[6] Kenton Varda. Protocol buffers: Google's data interchange format [EB/OL]. [2008-06-24]. http://google-opensource.blogspot.com/2008/07/protocol-buffers-googles-data.html.

Application and Research on Parsing Techniques of Large XML Document

ZHANG Tai-biao[1], ZENG Wen-hua[1*], CHEN Zhi-wei[2]

(1. School of Software, Key Laboratory of Intelligent Information Technology of Fujian Province, Xiamen University, 2. School of Physics and Mechanical & Electrical Engineering, Xiamen University, Xiamen 361005, China)

Abstract: With the continuous increase of the XML applications, the research of XML quickly parsing techniques have become a hot topic for current research. The paper mainly concerned the parsing and processing of the large XML document. Firstly, it discussed the techniques of XML parsing which fit for large XML document at present, such as, SAX and VTD-XML, it also analyzed and compared their respective advantages and disadvantages. Secondly, it made a solution to the question of XML processing encountered on project of NMR, compared the advantages and disadvantages of the two solutions, then draws a conclusion that both of them fit for the project. Furthermore, according to the faults of the two solutions mentioned previous; it proposed some other techniques fitting for large document.

Key words: large XML document; XML parser, SAX; VTD-XML