计算机科学 2005Vol. 32№. 11

面向压缩 XML 文档的数据快速过滤与片断分发方法*)

吴 泠 杨冬青 唐世渭 王腾蛟 包小源
(北京大学信息科学技术学院 北京 100871)

摘 要 XML 数据的过滤和分发是 XML 应用领域的研究热点之一。为了提高 XML 文档的传输效率,有必要进行压缩。本文提出一种应用于压缩 XML 文档的过滤和片断分发方法。实验表明,这种方法既保持了压缩文档的传输效率优势,又滤掉了非查询目标数据,提高了分发中心的匹配/分发处理效率。

关键词 XML,算术压缩,滤波,分发,Bloom 滤波器

Efficient Filtering and Fragment Dissemination of Compressed XML Documents

WU Ling YANG Dong-Qing TANG Shi-Wei WANG Teng-Jiao BAO Xiao-Yuan
(School of Electronics Engineering and Computer Science, Peking University, Beijing 100871)

Abstract The filtering and dissemination of XML documents is one of the most popular research areas of XML applications. In order to improve the transfer efficiency of XML documents, compression should be used. But few of the filtering and dissemination algorithms of compressed XML documents have been proposed. We introduce a new method of filtering and dissemination of compressed XML document, which can efficiently filter and accurately disseminate the fragments to end users. Experiments show that our method is time and space efficient.

Keywords XML, Arithmetic compression, Filtering, Dissemination, Bloom filter

1 引言

XML 数据的过滤及分发是目前 XML 数据管理领域研究的热点之一[2~4]。近来互联网的发展产生了一种新的应用 SDI(Selective Dissemination of Information),即信息的选择性分发。由于 SDI 应用要求系统能够对互联网上的海量 XML 文档进行基于内容的实时过滤分发,因此如何进行高效的处理、匹配以及分发便成为一个重要问题。

作为数据交换主要格式,XML 文档需要在网络中大量传输。XML 包含许多冗余信息,为提高网络传输效率,应采用压缩 XML 文档的方法。

目前针对 XML 文档提出了很多过滤方法[2~4],但上述方法均基于非压缩 XML 文档,而无法处理压缩 XML 的情况。并且这些方法都是将与用户查询相匹配的文档整个发送给用户,而这些文档中也同时包含许多与用户查询无关的信息,因此增加了分发中心的负荷,降低了分发精确度。

本文提出一种应用于压缩 XML 文档的过滤和片断分发方法。该方法可直接应用于压缩 XML 文档,既能保持压缩文档的传输效率优势,又能滤掉非查询目标数据,提高分发中心的匹配/分发处理效率。具体研究成果包括:1 提出了一种基于压缩 XML 文档的分析和过滤方法,可实现压缩 XML 文档与用户查询的高效匹配;2 提出了压缩 XML 文档的片断分发方法,实现了按照用户需求的精确分发;3 采用标准数据进行实验和对比分析,实验结果验证了该方法的有效性。

文章第 2、3 节将对相关工作、问题定义及相关概念进行介绍;第 4 节介绍面向压缩 XML 的快速过滤及片断分发机制;第 5 节简要介绍过滤系统的实现以及实验结果。最后是总结及未来工作展望。

2 相关工作

目前对 XML 文档过滤提出的方法主要包括两种:使用自动机[2,4]及索引[3]的方法。使用自动机的方法较有代表性的为 YFilter[2]。YFilter 使用一个 NFA(Nondeterministic Finite Machine)来表示所有的用户查询语句,这样在对具有公共前缀的语句进行处理时,可以共享很多处理步骤。使用索引的方法较有代表性的是 Index-Filter[3],它使用前缀树(Prefix Tree)来表示所有的查询语句,并对到来的 XML 文档元素进行索引。

但这些方法均针对整个文档进行,即分发中心将匹配文档整个发送给相应用户,这样就发送了许多与用户需求无关的冗余信息,增加了网络及分发中心的负担。且这些方法均针对非压缩的 XML 文档进行,而对压缩 XML 文档鲜有相关研究。

本文提出了一种对压缩 XML 文档片段进行过滤分发的机制,既能保持压缩文档的空间优势,又能滤掉冗余信息实现准确分发。

3 问题定义及相关概念

3.1 问题定义

由于 XML 文档结构中存在大量冗余信息,因此传输压缩后的文档可以大大减少网络传输时间。但是,如何使用压缩文档来与用户请求进行匹配,便成为一个重要问题。因此,下面提出压缩 XML 文档的过滤问题定义:

定义 1(压缩 XML 文档的过滤) 给定用户集 $U=\{u_1, u_2, \cdots, u_n\}$,对其中任意 $u_i \in U(1 \leqslant i \leqslant n)$,有 $Q_i=\{q_{i1}, q_{i2}, \cdots, q_{im}\}$,其中 q_{ij} $(1 \leqslant j \leqslant m)$ 为路径查询语句。对到来的压缩

*)本文研究得到国家 863 数据库重大专项课题(2005AA4Z3070)和国家自然科学基金项目(60473051)支持。

XML 文档 c,元素集 $E=\{e_1,e_2,\cdots,e_k\}$ 为压缩前 XML 文档包含的所有元素。若对于任意用户 u_i 的路径查询语句 q_{ij},有 c 中的元素 e_l 与 q_{ij} 相匹配,则将 e_l 对应的压缩 XML 文档 c 的片段 p 发送给用户 u_i。以上过程即为压缩 XML 文档的过滤。

压缩 XML 文档与用户查询的匹配通常有两种方法:一是将文档解压,再将还原出的文档与用户请求进行匹配。但由于文档在网络中通常要经过多级发布者的传递,因此如果每个文档都解压匹配之后再压缩发送,将给分发中心造成极大负担。二是直接用压缩文档与用户请求进行匹配。这种方法由于省却了压缩和解压的过程,使得处理和传输的效率大大提高。由于要直接在压缩文档上进行匹配,因此必须使用能保持 XML 文档结构的压缩方法。

3.2 相关概念

• *节点路径及路径查询语句* 可以采用一种半结构化数据模型 OEM[6](Object Exchange Model)来描述 XML 数据内容,称 XML 数据树。

定义 2(节点路径) 节点路径 $P=v_1/v_2/\cdots/v_m$。其中 v_i 为从 XML 数据树的根节点开始到达该节点为止依次经过的有向边。根结点用"/"表示。

有很多 XML 查询语言使用节点路径的方式来标识待查元素,比如 XPath,XQuery 等。本文采用路径的方式来表示元素的位置。

定义 3(路径查询语句) 一个路径查询语句 $Q=a_1e_1a_2e_2\cdots a_ne_n$,其中 a_i 为轴符号,e_i 为元素名。元素名包括文档中元素的标记以及通配符"*";轴符号包括表示父子关系的轴"/"以及表示祖先后代关系的"//"。为叙述简单起见,本文在后边的算法中只考虑不带"*"的查询语句,包括"*"的查询语句的处理可通过扩展得到。

• *结构保持的 XML 数据压缩* 为了使分发中心在过滤时能够清晰分辨出压缩 XML 文档中的标记和数据值,需要使用结构保持的 XML 数据压缩方法——反向算术压缩[1],该方法最大特点是将标记与数据值分开进行压缩,使得它们在压缩流中极易区分。

算术压缩的基本思想是将需要压缩的信息根据出现频率计算成区间值[min,max),不同符号的区间是不相交的。反向算术压缩是将[0.0, 1.0)区间分割成子区间,每个子区间对应 XML 文档中一个单独元素,区间大小与元素在文档中的出现频率相对应,使用元素的节点路径来计算区间值。

• Bloom Filter 使用一个二进制向量来保存集合中的所有元素。二进制向量的长度为 m,即向量共有 m 个二进制位,初始化时将所有位置为 0。假设有 n 个元素的集合 $S=\{s_1,s_2,\cdots,s_n\}$。对集合中的每一个元素 s_i,使用 k 个值域为 1 至 m 的散列函数 $H_1,\cdots,H_k$ 来将该元素散列到二进制向量上,即将 $H_j(s_i)$ 置为 $1(1\leqslant i\leqslant n,1\leqslant j\leqslant k)$。

若要检查元素 x 是否属于集合 S,则将散列函数 $H_1,\cdots,H_k$ 分别作用于 x,并检查 $H_j(x)$ 对应的位是否为 1。如果其中至少有一位为 0,则 x 不属于 S。

当然,Bloom filter 不是完全准确的,但是它只会将不属于集合 S 的元素判断为属于 S,而不会将本属于 S 的元素排除出集合。通常来讲,这种错误的概率非常小,通常为 0.5%,并且与 k,m 的取值有关。

4 面向压缩 XML 的快速过滤及片断分发机制

过滤及分发过程如下:用户将感兴趣的信息使用路径查询语句描述,提交给分发中心;分发中心收集路径查询语句,根据文档 DTD 将语句展开成完全节点路径,并使用反向算术压缩方法计算,将其成压缩区间值,并将该值加入 Bloom filter。压缩 XML 流到来时,顺序过滤该流。若遇到元素开始标记的压缩值,则检查其是否属于 Bloom filter。若属于,则存在与该元素对应的用户查询,找到对应用户,并将对应元素的压缩文档片段发送给用户。

4.1 DTD 结构图及 DTD 元素表

XML 文档 DTD 可用图来表示(称 DTD 结构图),是记录 XML 元素之间关系的数据结构。我们使用 DTD 元素表对 DTD 结构图进行存储,以便对查询语句进行展开。

定义 5(DTD 元素表) DTD 元素表是 DTD 图的存储形式,表中每项表示 DTD 图中的一个节点。表项由三个属性组成,即(ID, Parent, Name)。其中 ID 是图中节点的标识,Parent 是该节点的父节点,Name 是该节点表示的元素在文档中的标记。DTD 元素表使用 ID 和 Name 来存储结构图中的节点,使用 Parent 属性来存储节点之间的关系,也即文档中元素之间的父子关系。

4.2 用户查询语句的存储

使用 Bloom filter 存储查询主要分为两个步骤:首先是将路径查询语句展开成所有可能的节点路径,然后将这些节点路径使用算术压缩方法计算为一个压缩值区间,加入 Bloom filter 中。

由于路径查询语句中可以使用祖先-后代轴"//"来代替节点路径中的某些元素,语句所表达的路径信息是不完整的,因此这样的一条语句可能对应多条节点路径,也就表示了多个元素。在使用 Bloom Filter 进行过滤的时候,只有将所有可能元素对应的压缩值区间都加入 Bloom filter 中,才能够保证过滤匹配的正确性和完整性。

下面介绍查询语句转换算法。在 DTD 的基础上,将不完整的包含"//"的查询语句展开成一条(或多条)完全节点路径,即找到查询语句表示的所有可能的节点路径,为建立 Bloom filter 做好准备。

算法 1 路径查询语句展开(不包含"*"的语句)

输入:包含"//"的路径查询语句 $Q=a_1n_1a_2n_2\cdots a_ln_l$,

文档的 DTD 元素表 ET

输出:展开后得到的节点路径集合 FP

1. 初始化 FP 为空;
2. 在 ET 中查找所有 Name 属性值为 n_l 的表项,加入集合 CurrElement;
3. 若 CurrElement 为空,则没有找到对应的元素,匹配失败,算法结束;
4. 反向遍历 Q,对 $i([l..1]$;
5. 若 a_i 为"/",则从 ET 找到 CurrElement 所有元素;
6. 对其中每个元素,若它的父元素的 Name 属性与 n_i 不相等,则将该元素从 CurrElement 中删除;
7. 转 4;
8. 若 a_i 为"//",则从 ET 找到 CurrElement 中所有元素;
9. 对其中每个元素,从该元素起沿 Parent 属性向祖先元素遍历;
10. 如果遇到 Name 属性为 n_{i-1} 的元素则停止,将 CurrElement 中的该元素更新为最后遍历到的元素,转 4。否则,若遇到 Parent 属性为空的根节点,将该元素从 CurrElement 中删除;
11. CurrElement 中的元素节点路径集合即为 FP。

例如,使用算法 1 对路径查询语句/Teachers//City 进行展开,可以得到完全节点路径/Teachers/Teacher/Address/City。

使用算法 1 将路径查询语句展开成完全的节点路径,这是建立 Bloom filter 的第一步。下面还需要将每条节点路径放入 Bloom filter 中。此时使用反向算术压缩[1]方法,对节点路径进行压缩,生成压缩值区间[min, max),再将 min 加入

到 Bloom filter 中。对所有的节点路径逐个处理，当所有的语句都处理完毕，则 Bloom filter 建立成功。

4.3 查询语句的更新策略

由于路径查询语句中可以使用祖先-后代轴“//”，因此它可以表示多条节点路径。同样地，多条路径查询语句也可以表示同一条节点路径，也就会被压缩得到相同的压缩值区间。相同的压缩值在 Bloom filter 中只需加入一次。由于路径查询语句可能来自不同的用户，因此对于 Bloom filter 中的每一个压缩值，都需要记录有哪些用户查询与之对应，这样才能够将文档片段正确发送。

另外，在用户查询变化时，Bloom filter 也要进行相应更新，删除其中不会再使用的查询语句，加入新的用户请求。在删除某条查询语句的时候，将对应区间值的计数减 1。若计数变为 0，则将该区间值删除。增加查询语句时，首先查找该语句是否已经在 Bloom filter 中。若已存在，则计数加 1，否则将对应的压缩区间值加入，并将计数设为 1。

我们使用压缩值区间倒排表来存储用户查询与压缩区间值的对应关系及为压缩区间值进行引用计数。下面叙述压缩区间值倒排表的构造。

定义 6(压缩区间值倒排表) 压缩区间值倒排表包含若干入口(Entry)，每个入口记录了一个压缩区间值的下界。入口包含一个线性表，表中记录了该区间值对应的用户查询的标识。图 1 为压缩区间值倒排表的示例。

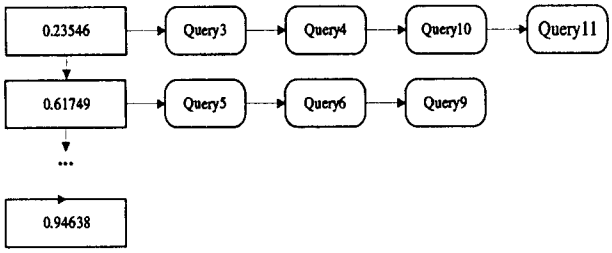

图 1 压缩区间值倒排表示例

当压缩文档中的标记与 Bloom filter 中的压缩区间值相匹配时，便可以该值为入口，从倒排表中找到对应的所有用户查询，并以此发送给相应的用户，从而保证了分发的正确性。

4.4 压缩 XML 文档片段分发方法

我们使用 Bloom filter 对用户查询语句进行存储后，为压缩 XML 文档到来做好了准备。下面介绍片段分发方法。

本文提出的方法支持对压缩文档的全文分发以及片段分发。对于全文分发，只需从 XML 文件流中找到元素开始标记的压缩值，检查其是否在 Bloom filter 中。若发现元素开始标记属于 Bloom filter，则从压缩值区间倒排表中找到该值对应的所有查询语句，即可找到匹配的用户。再将取到的完整文档发送给对应用户即可。片段分发需要分辨出元素对应的每一个片段的开始及结束位置，因而相对复杂。但 XML 文档的标记(Tag)都是成对出现的，故可使用栈来记录每次处理的标记。算法 2 是对该分发过程的描述：

算法 2 压缩 XML 文档的片断分发

输入：压缩 XML 文档流

输出：各终端订户对应文档片断

1. 初始化标记栈 TagStack；
2. while 文档未结束；
3. 顺序读入流，直至发现元素标记的压缩值 tag；
4. 若 tag 为元素开始标记，则 push tag，并在 Bloom filter 中检查 tag 是否出现；
5. 若出现，记录此时栈的高度 depth；从压缩区间值倒排表中找到对应的查询及用户，开始分发；
6. 转 2；
7. 若 tag 为元素结束标记；
8. 若此时栈高度 depth 与某正在分发的线程相同，则该线程分发结束；pop tag；
9. 转 2。

该算法提出的压缩 XML 文档的片断分发方法同样也可以应用于非压缩 XML 文档。

5 实验结果

本节将本文所述方法与基于自动机的过滤算法 YFilter[2] 从三个方面进行比较。所有实验均在如下环境下完成：CPU-P3 800MHz，内存 256M，操作系统 Windows XP Professional。全部实验代码使用 Java 编写。实验数据使用 DBLP[7] 数据集。

采用 XPRESS 算法[1] 对实验数据进行压缩。为更好地与 YFilter 进行比较，仅对原 XML 文档中的元素标记(Tag)进行压缩，而没有压缩元素的数据值。在仅对元素标记进行压缩的情况下，使用 DBLP 数据集的平均压缩率达到 30.2%，可见压缩对于提高 XML 文件的存储效率是非常有效的。平均情况下使用 XPRESS 对 XML 文档的所有内容进行压缩的压缩率可达 73%[1]。

5.1 存储结构的建立

本节通过实验对本文中建立 Bloom filter 存储结构所需时间与 YFilter 中 NFA 的建立时间进行比较。建立 Bloom filter 存储结构所需时间包括查询语句的展开以及 Bloom filter 的建立两部分。图 2 与图 3 是在不同文档深度与不同查询语句数量下两种方法存储结构建立时间的比较。

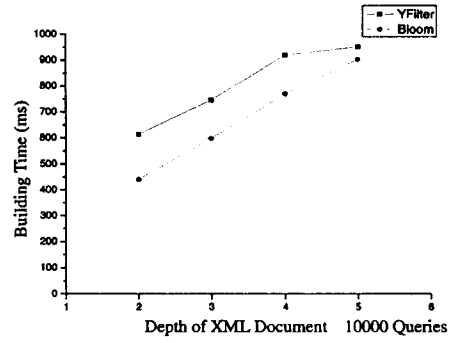

图 2 不同文档深度下存储结构建立时间

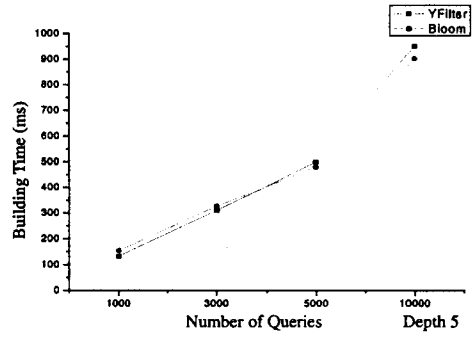

图 3 不同语句数量下存储结构建立时间

实验结果表明，在查询语句数量一定的情况下，本文的存储结构建立时间低于 YFilter 的建立时间。同时，两种方法的

建立时间都随文档深度的增加而增大，本文方法增加趋势更大。而在文档深度一定时，两种方法的建立时间都随查询语句数量增多而增大，但 YFilter 增长趋势更快。当语句达到5000 条以上时，YFilter 的建立时间超过本文方法。这是由两种存储结构的构造方法决定的。对于 YFilter，首先需要对所有查询语句建立前缀树，然后再逐条语句加入 NFA 中。当查询语句数量增大时，NFA 中状态显著增多，因此对查询语句的数量变化较为敏感。对本文 Bloom filter 方法，由于首先需要将路径查询语句展开成完全节点路径并压缩成区间值，当节点及查询语句深度增大时展开和压缩时间增大，因而对文档深度变化更加敏感。

5.2 压缩 XML 文档的过滤时间

本节对本文中 Bloom filter 存储结构与 YFilter 中 NFA 对 XML 文档的过滤时间进行比较。由于本文所述方法是直接作用于压缩 XML 文档上，因而对 YFilter 我们采取先解压再过滤的方法。图 4 与图 5 是在不同文档深度与不同查询语句数量下两种方法对压缩 XML 文档过滤时间的比较。

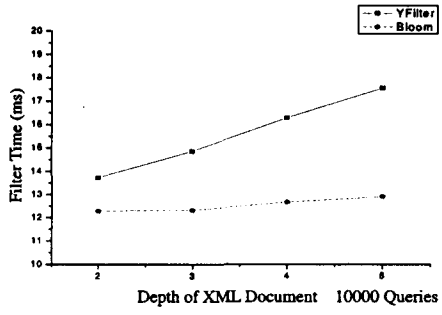

图 4 不同文档深度下压缩文档过滤时间

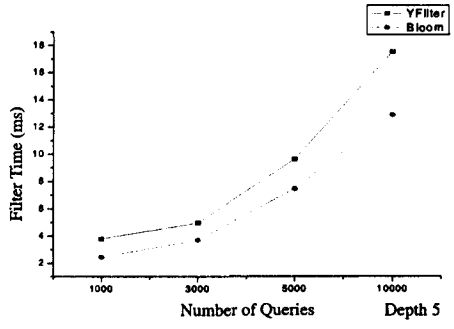

图 5 不同语句数量下压缩文档过滤时间

可以看到，在查询语句数量一定的条件下，两种方法处理时间都随文档深度增加而增长，其中 YFilter 更加明显。在文档深度一定时，查询语句数量对 YFilter 影响更大。这是由两种方法进行匹配的基本原理决定的。由于 YFilter 使用 NFA 来进行匹配，当用户查询数量增大时，NFA 中状态增多，匹配时需要处理更多状态，故对查询数量变化较为敏感。对本文方法匹配时，只需将散列函数作用于压缩区间值，处理时间与查询数量无关。查询语句数量仅影响使用压缩值从压缩区间倒排表中取得相应用户的过程。由此可见，当查询语句数量较大时，本文所述基于 Bloom filter 的过滤方法非常有效。

5.3 片断分发效率

本节通过实验测试片断分发对传输效率的提高。定义片断分发率 FDR(Fragment dissemination ratio)作为测试标准：FDR = 片断分发给所有用户的片断大小之和 / 全文分发给所有用户的文档大小之和。实验结果表明，不同文档深度下片断分发率 FDR 有较小变化；在不同查询语句数量下，FDR 基本稳定。这是由于当查询语句近似随机产生时，命中文档中各元素的概率近似相等，因而片断分发率与文档的结构有关，与查询语句数量无关。对 DBLP 数据，FDR 基本稳定在 22%左右，因此使用片断分发占用的网络传输资源仅为全文分发的 22%左右，能够很好地降低网络传输负载。

由以上三节实验可见，本文提出的基于压缩 XML 文档的过滤及片断分发方法是有效的。

结论及后续研究 本文提出一种直接基于压缩 XML 数据流的过滤方法，并可有效地根据用户查询来进行 XML 压缩文档的片段分发。本文所述方法既可直接作用于压缩 XML 文档，又能够滤掉文档中的冗余信息，将所需片段精确发送给用户，缩短了文档的网络传输时间，减轻了分发中心的负荷，提高了分发准确率。

后续工作将讨论如何更完整地支持路径查询语句(例如在查询语句中加入对谓词的支持)，以及改进算法，使其进一步提高运行效率，使其能够对更大量的用户查询语句进行更高效的处理。

参 考 文 献

1 Min J-k, et al. XPRESS: A queriable compression for XML data. In: ACM SIGMOD, 2003. 122～133

2 Diao Yanlei, Altinel M, Franklin M J, et al. Path sharing and predicate evaluation for high-performance XML filtering. ACM Transactions on Database Systems (TODS), 2003, 28(4): 467～516

3 Brumo N, Gravano L, Koudas N, et al. Navigation- vs. index-based xml multi-query processing. In: ICDE, 2003. 139～150

4 Chan C, Filber P, Garofalakis M, et al. Efficient filtering of XML documents with xpath processing. In: ICDE, 2002. 235

5 Bloom B H. Space/time trade-offs in hash coding with allowable errors. Communications of the ACM, 1970, 13(7): 422～426

6 Papakonstantinou Y, Garcia-Molina H, Widom J. Object Exchange Across Heterogeneous Information Sources. In: Proc. of the 11th Int'l Conf. on Data Engineering (ICDE' 95). Washington: IEEE Computer Society, 1995. 251～260

7 DBLP, XML Data. At http://dblp.uni-trier.de/xml/