一种适合分布式虚拟环境的XML路由技术*)

陈继明[1,2] 贝 佳[2] 徐 丹[2] 鞠时光[1] 潘金贵[2]
(江苏大学计算机科学与通信工程学院 镇江 212013)[1]
(南京大学计算机软件新技术国家重点实验室 南京 210093)[2]

摘 要 在分布式虚拟环境中,XML技术的应用提高系统的实用性和扩展性。本文针对分布式虚拟环境系统中的XML路由问题,提出一种新的XML路由技术—BPfilter。该技术根据分布式虚拟环境的特点,在Bloom过滤器算法的基础上,引入语法树以处理XPath查询中判定词,实现了对系统中XML数据快速地匹配和路由。实验结果表明,该技术是可行和有效的。

关键词 分布式虚拟环境,XML路由,Bloom算法,BPfilter

An Adaptive XML Routing Technique in Distributed Virtual Environment

CHEN Ji-Ming[1,2] BEI Jia[2] XU Dan[2] JU Shi-Guang[1] PAN Jin-Gui[2]
(School of Computer Science and Telecommunications Engineering, Jiangsu University, Zhenjiang 212013)[1]
(State Key Lab for Novel Software Technology, Nanjing University, Nanjing 210093)[2]

Abstract The application of XML in distributed virtual environment gives a practical way to enhance the practicability and scalability of the system. In order to increases the rate of XML routing in the system, BPfilter which is a new XML routing technique is given in this paper. According to the characteristic of distributed virtual environment, this method is extended bloom filter-based algorithm with syntax tree to support predicators in XPath queries, and can perform matching and routing of XML data in the system. Finally, this technique is verified to be efficient and effective by experiment results.

Keywords Distributed virtual environment, XML routing, Bloom algorithm, BPfilter

传统分布式虚拟环境的主动兴趣管理系统中,参与者采用由多个属性的值或值域组成的兴趣表达式来发布数据包,如AIMNET系统[1],每个属性是一个三元组,包括属性值的类型、属性名称和属性值(或值域)。同一个兴趣表达式中多个属性条件是逻辑与的关系,只有满足所有属性条件的数据才是属于兴趣表示范围内的数据,而不同兴趣表达式之间是"或"关系。兴趣表达式虽然在一定程度上提高了对象的兴趣表达力,但在实现时,通常将参与主动兴趣管理的兴趣属性放置在数据包的固定位置的方式,不仅限制了对象兴趣表达能力,且依赖于特定应用,同时在主动兴趣管理过程中需要定义固定的传输协议格式并实现相应的协议解析和过滤算法,不具有通用性,因此很难在网络上为分布式虚拟环境系统的应用设计和部署实际的主动路由器。

将XML路由技术[2]应用于分布式虚拟环境后,XML文档的层次式树状结构和XPath规范中的通配符以及轴(axes)使得参与者发布的兴趣属性只要被组织成合适的XML文档片断,就能够被放置在XML数据包的任何位置。只要遵循XML和XPath规范,就能完成对XML数据包的路由,从而在XML路由网络上构造基于主动兴趣管理的分布式虚拟环境系统,将极大增强主动路由技术的实用性,有效地提高分布式虚拟环境系统的可扩展性。

针对XML数据包的XML路由技术,本质上就是XPath查询的匹配。XML路由器进行路由时,为了处理XML文档的层次式树状结构和XPath规范的强大表示能力,需要大量的额外开销,同时XPath查询的匹配时间随订购的数量呈指数或多项式增长[3]。随着分布式虚拟环境的规模和参与者的增长,XML数据和XPath查询的数量也势必不断增长,XML路由器必然成为整个分布式虚拟环境系统的瓶颈。因此,针对XML路由技术的研究对于提高XML路由器对XML数据包进行路由的速度,从而提高基于XML路由的分布式虚拟环境系统的可扩展性有着重要得意义。

1 分布式虚拟环境中的XML路由问题

分布式虚拟环境中参与者个数众多,并且参与者同时担任发布者和订购者的角色,在基于XML路由的分布式虚拟系统需要匹配的XML数据和XPath查询的数量相当可观。因此,与注重匹配效率的应用相比,在分布式虚拟环境中使用的XML路由技术该算法具有以下特点:(1)能够快速高效地处理和路由XML数据信息;(2)能够实现快速地处理XPath查询信息的更新。

结合上述特点,基于Bloom过滤器的XML路由技术[4]能根据XML流产生的SAX事件生成候选路径表达式,通过

*)本课题得到国家自然科学基金项目(60473113)、国家自然科学基金重点项目(60533080)、江苏首高校自然科学指导性计划项目(05KJD520051)资助。**陈继明** 博士研究生,主要研究领域为XML、分布式虚拟环境;**贝 佳** 博士研究生,主要研究领域为虚拟环境、Agent技术;**徐 丹** 硕士研究生,主要研究领域为虚拟环境;**鞠时光** 教授,博士生导师,主要研究领域为数据库、XML;**潘金贵** 教授,博士生导师,主要研究领域为多媒体信息处理、多媒体远程教育系统。

散列的方式和 XPath 查询比较。与基于自动机的 XPath 查询匹配算法[7,8]相比，Bloom 算法的时间和空间复杂度均可以控制在常数级别，较适合于分布式虚拟环境应用。但是在采用 XPath 查询来表示兴趣表达式时需要使用判定词来表示限定条件，而目前的 Bloom 过滤器的 XML 路由技术通过产生候选路径表达式并采用字符串匹配的方法是无法处理判定词的，因此如果将基于 Bloom 过滤器的 XPath 查询匹配算法引入基于 XML 路由的分布式虚拟环境中，就需要解决如何处理判定词这一问题。

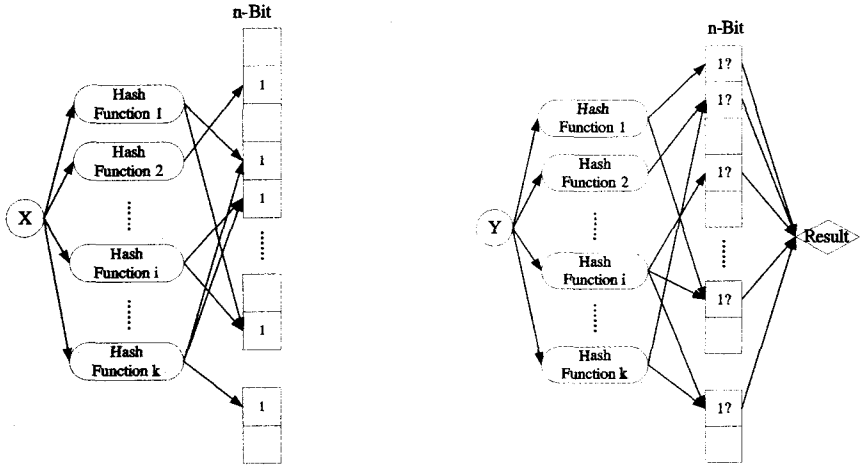
2 Bloom 过滤器

由于其良好的时间和空间复杂度，作为一种用以表示支持关键字查询的集合的数据结构，Bloom 过滤器[5]被广泛应用于数据库应用，Web 缓存、侵入检测、查询过滤和路由等多种应用[9,10]。

Bloom 过滤器本质上是一个 m 维字位向量，在向 Bloom 过滤器加入对象或检查 Bloom 过滤器是否包含特定对象时，需要使用事先选定的 k 个哈希结果同为 m 维字位向量的哈希函数。图 1 描述了一个 Bloom 过滤器。

初始时，Bloom 过滤器的所有字位均被置为 0。当需要将对象加入 Bloom 过滤器时，使用选定的 k 个哈希函数对该对象进行哈希操作。如果哈希结果中的第 i 位为 1，那么将 Bloom 过滤器的第 i 位也置为 1。在处理完所有的 k 个哈希结果后，就完成了对象的加入工作。在检查 Bloom 过滤器是否包含特定对象时，仍然使用同样的 k 个哈希函数对该对象进行哈希操作，如果这 k 个哈希结果中所有被置为 1 的位，在 Bloom 过滤器中对应的位也同样被置为 1，那么可以判定当前的 Bloom 过滤器包含了该对象。反之，当前的 Bloom 过滤器不包含该对象。

由于 Bloom 过滤器使用字位向量来表示对象集合，并使用按位与加入对象，因此一旦将对象加入 Bloom 过滤器后，就无法再删除。针对这种情况，计数 Bloom 过滤器(Counting Bloom Filtering)为字位向量中的每个字位都设置一个计数器[6]。在加入对象时，如果哈希结果的第 i 位为 1，在设置 Bloom 过滤器的第 i 位为 1 的同时，也递增该字位对应的计数器。同样地，如果需要删除对象时，也需要使用 k 个哈希函数进行哈希操作。如果哈希结果的第 i 位为 1，则递减该字位对应的计数器；如果该计数器已经归零，则将 Bloom 过滤器的第 i 位置零。在处理完所有 k 个哈希结果后，即完成对象的删除工作。显然，在 XML 路由技术中，路由表不仅需要拥有加入 XPath 查询的能力，也需要删除已有 XPath 查询的能力，一般选择计数 Bloom 过滤器为主要数据结构。



(a) 加入对象　　(b) 检查对象 Y

图 1　Bloom 过滤器

3 BPfilter 路由技术

在目前基于 Bloom 过滤器的 XML 路由技术中，由于根据 SAX 事件无法一一枚举所有的判定词，无法生成带有判定词的候选路径表达式，需要使用其他的方式完成对判定词的表示和处理。因此，本文提出了一种支持判定词的 XML 路由技术－BPfilter，使用了语法树表示判定词，将判定词中的路径表达式视作变量，在匹配 XML 数据时，使用 Bloom 过滤器完成对这些变量的值的收集，完成语法树的计算，并根据计算结果控制 XPath 查询的匹配。

3.1 XPath 查询规范子集

我们首先定义 BPfilter 支持的 XPath 规范的子集(如图 2 所示)，其中 lable 是合法的 XML 元素标签，Const 则是合法的常量。XPath 查询规范子集，支持子轴、后代轴以及判定词。在判定词中，不仅支持比较操作和布尔操作，还支持算术操作。因此，该 XPath 查询规范子集已经能够初步满足分布式虚拟环境对 XPath 查询匹配的要求。

```
P ::= /T | //T
T ::= label | text() | * | T/T | T//T | T[Q]
E ::= label | text() | * | E/E | E//E
Q ::= E | Q Oprel Const | Q Oprel Q | Const Oprel Q | Q and Q | Q or Q | not(Q)
Oprel ::= < | <= | > | >= | = | != | + | - | * | div
```

图 2　XPath 查询规范子集

此外，图 2 中的 XPath 查询规范子集仅仅涵盖了对 XML 元素的查询，并不支持对 XML 数据中的属性进行查询。事实上，除了不能包含子元素以外，XML 数据中的属性同 XML 元素是类似的，XPath 查询中对 XML 元素和属性的查询也是类似的。本文为了叙述的统一，省略了对属性的支持，即在 XPath 查询匹配算法中对 XML 元素查询和对属性查询的采用一致的处理方法。

3.2 数据结构

图 3 显示了 BPfilter 的主要数据结构。除了路由表和前缀过滤器外，还包括 XPath 查询表、判定词表、语法树、变量

表和结果缓存。其中，语法树表示了判定词的计算逻辑；XPath 查询表则维护了特定 XPath 查询的简单 XPath 查询和属于该 XPath 查询的语法树的指针；变量表则维护了所有判定词中的路径表达式，并为语法树所引用；结果缓存则提供了匹配过程中简单 XPath 查询和 XML 元素匹配结果的存储。

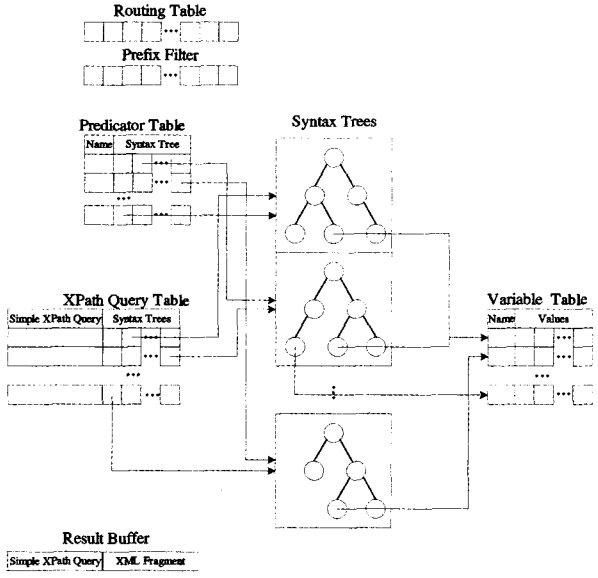


图 3　数据结构

如果将一个长度为 k 的 XPath 查询 q 记作 $s_1\ s_2 \cdots s_k$，对于第 i 个步进 s_i，该步进被记作 $a_i n_i c_i$，其中 a_i 是代表子轴的“/”或者代表后代轴的“//”，n_i 是 XML 节点检测或通配符“*”，c_i 是修饰 n_i 的判定词。如果该步进不包含判定词，则 c_i 为空字符串，记作 $c_i=\emptyset$；如果该步进包含判定词，则 c_i 不为空。如果去除所有的判定词，那么 XPath 查询 q′就可以表示 $a_1 n_1 a_2 n_2 \cdots a_k n_k$，q′被称作 q 对应的简单 XPath 查询。

如果 XPath 查询带有判定词，在同 XML 数据进行匹配时，不仅要求 XML 数据和 XPath 查询对应的简单 XPath 查询相匹配，XML 数据还需要满足 XPath 查询中的所有判定词。路由表维护了 XPath 查询表达式对应的简单 XPath 查询，它反映了 XPath 查询中的结构匹配。针对判定词表的检查结果标志了一个判定词作用域的开始和结束。由于在多个 XPath 查询中，相同的路径表达式可能拥有不同的判定词加以修饰，因此在一个表项中可能拥有一个或多个语法树的引用。

3.3　路由过程

在加入一个长度为 k 的 XPath 查询时，需将其对应的查询 $a_1 n_1 c_1, a_2 n_2 c_2, \cdots, a_k n_k c_k$ 加入路由表中，即在 XPath 查询表中新建一项。对于第 i 个判定词 c_i，如果 c_i 不为空，首先，根据 c_i 构造语法树，并将其索引加入该 XPath 查询表中对应的表项；然后，将其路径表达式 $a_1 n_1, a_2 n_2, \cdots, a_k n_k$ 加入判定词表；最后，还需要根据 c_i 中的所有路径表达式，增加变量表中的表项。同时为了控制候选路径表达式的数量，不仅需为前缀过滤器加入简单 XPath 查询的前缀，还需要加入判定词中路径表达式的前缀。

值得注意的是，在这样的结构中，如果需要加入的语法树或者变量已经存在，则表示在已经加入的 XPath 查询中已经包含了相同的判定词或者判定词中的路径表达式，就可以直接加以引用而不必重复加入。因此，在 BPfliter 路由技术中，不仅能够发现和共享公共判定词的处理，还可以实现对判定词中的公共路径表达式的处理。

下面我们以图 4 中的 XPath 查询为例，对上述加入过程进行阐述。在此 XPath 中 k 为 4，其中 c_2, c_3 为非空字符串。首先为路由表加入“//a/b/e//g”；接着对于 XPath 包含的每个非空判定词 c_2, c_3，构建其构造语法树，并将其索引加入该 XPath 查询表中对应的表项。最后将其修饰的简单路径“//a/b”和“//a/b/e”加入。图 5 显示了处理图 4 中的 XPath 查询后，相关数据结构的状态。

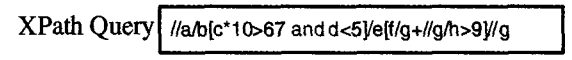


图 4　XPath 查询表达式

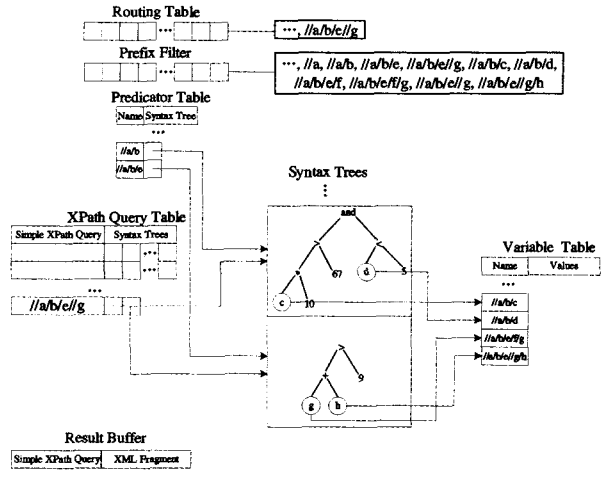


图 5　加入 XPath 查询的数据结构

基于上述数据结构的 XML 数据和 XPath 查询的匹配，在处理 XML 数据产生的 SAX 事件时，不仅需要使用产生的候选路径表达式检查路由表，还需要检查判定词表和 XPath 查询表，主要步骤如下：

1）在处理 startElement 事件时（标志了处理 XML 数据元素的开始），如果判定词表包含了某个候选路径表达式，那么在当前元素未结束前，均是该判定词的作用域，并可能存在简单 XPath 查询对其子元素进行查询。只有当这种简单查询和 XML 元素相匹配，且判定词为真时，才能判定 XPath 查询和 XML 元素匹配。因此，在判定词的作用域结束前，使用结果缓存来记录匹配的简单 XPath 查询和 XML 元素。在判定词的作用域中，还需要处理 characters 事件，根据生成的候选路径表达式检查变量表，如果存在这样的表项，则将当前文本作为变量值填入变量表。

2）在处理 endElement 事件时（标志了处理 XML 数据元素的结束），除了完成对候选路径表达式的维护外，也需要检查判定词表。如果判定词表中包含了以某个候选路径表达式为名的项，那么该 SAX 事件标志着这些判定词作用域的结束，需要对这些判定词对应的语法树进行计算。如果语法树引用的变量中有空变量，那么该语法树为假；否则使用变量表中的相关变量的所有值，计算语法树的结果，如果计算结果存在真值，该语法树为真，否则，该语法树为假。在完成语法树的计算后，除了将其引用的所有变量置为空外，还需遍历 XPath 查询表。如果某项中引用的对应的语法树均为真，且结果缓存中包含了该项中的简单 XPath 查询，那么结果缓存

中同 XPath 查询匹配的 XML 元素也同该项表示的 XPath 查询匹配。

4 实验

为了验证本文提出的 XML 路由算法的性能，我们在 AIMNET 系统平台的基础上实现了一个基于 XML 路由的通讯结构的原型系统，即通过在工作站上运行软件模拟 XML 路由器的功能来实现，其工作站配置为：CPU Xeon 2.8，RAM 1G，System Windows XP Professional。

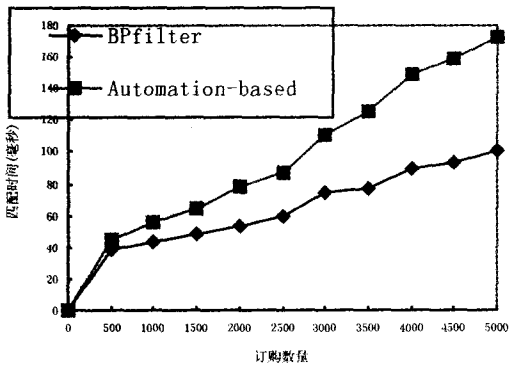


图 6 匹配时间图

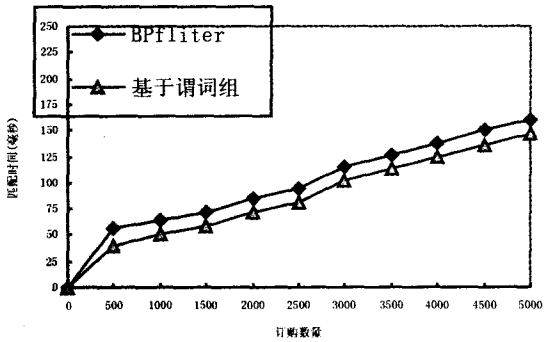


图 7 匹配时间图

图 6 和图 7 分别描述了 BPfliter 路由算法与基于自动机和基于谓词组的路由算法的相关实验性能数据。从图 4 中可以看出，在分布式虚拟环境系统的应用下，BPfliter 路由算法的性能要明显优于基于自动机的路由算法，并且随着订购数量的增加，这种优势表现的越来越明显。从图 5 中可以看出，BPfliter 路由算法的性能稍低于基于谓词组的匹配算法，这主要是由于 BPfliter 中包含了 XML 数据的解析时间，在本实验路由器的配置中，解析一个实验用的 XML 数据需要 180μs 的时间，100 个 XML 数据即是 18μs，减去解析时间后，BPfliter 路由算法的性能的匹配时间则大致和基于谓词组的方法相当。

结束语 由于分布式虚拟环境具有参与者个数众多，信息更新快等特点，使得拥有良好时间和空间复杂度的基于 Bloom 过滤器的 XML 路由技术十分适合于分布式虚拟环境的应用。本文在基于 Bloom 过滤器的基础上，提出了一种支持判定词的 XML 路由技术，保留了使用候选表达式和前缀表达式来进行结构匹配的优点，采用语法树的结构表示 XPath 查询中的判定词，以变量的方式表示判定词中的路径表达式，扩展了基于 Bloom 过滤器的 XPath 查询匹配算法，使其初步具备应用于分布式虚拟环境系统的能力。

参 考 文 献

1 卢威，陈继明，徐晓旸，等. 分布式虚拟环境 AIMINET 的关键技术概述. 计算机科学，2006，33(11)

2 贝佳，翟磊，陈继明，等. 基于 XML 路由网络的主动兴趣管理研究. 计算机科学，2007，34(1)

3 Gottlob G，Koch C，Pichler R. Efficient algorithm for processing XPath queries. In：Proceedings of VLDB，2002

4 Gong X，Qian W，Yan Y，et al. Bloom Filter-based XML Packets Filtering for Millions of Path Queries. In：Proceedings of the 21st International Conference on Data Engineering，2005

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

6 Fan L，Cao P，Almeida J，et al. Summary cache：a scalable wide-area web cache sharing protocol. In：Proceedings of the ACM SIGCOMM '98 Conference on Applications. ACM Press，1998

7 Diao Y L，Fischer P，Fanklin M J，et al. Yfilter：efficient and scalable filtering of XML documents. In：Proceedings of ICDE，2002. 341～346

8 Altne M，Franklin M J. Efficient Filtering of XML Documents for Selective Dissemination of Information. In：Proc. of VLDB'00，2000

9 Mullin J. Optimal semijoins for distributed database systems. IEEE Transactions on Software Engineering，1990，16(5)

10 Fan L，Cao P，Almeida J，et al. Summary cache：a scalable wide-area web cache sharing protocol. In：Proceedings of the ACM SIGCOMM '98 Conference on Applications，Technologies，Architectures，and Protocols for Computer Communication，1998

(上接第 244 页)

- 改进了有关的扩展概念；增加硬件扩展方面的内容；
- 去除共同特性和高级实践；
- 去除了 SS 附加信息；ISM 并入 SAM；
- 阐明了“不适用的”过程域的指导原则；
- 改进了概述和术语表；
- 添加工作环境内容到 OPD 和 IPM 中；
- 简化、整理了 IPPD 的相关内容；
- 在 IPM 和 OPF 中强调了过程部署的重要性。

结束语 最新发布的 CMMI V1.2 模型做了很多的改进。将原先的模型 CMMI- SE/SW/IPPD 进行了整合，更改了模型的名称。CMMI- SE/SW/改为用于开发的 CMMI 模型 CMMI-DEV v1.2。CMMI v1.2 产品集支持开发、服务和获取过程，CMMI-DEV 是其中的一种并首先发布，其它的 CMMI 套件在 2007 年陆续发布。在 CMMI V1.2 版本中，模型的表述，术语，过程域都有相应的改进。本文概述这些变化，以期对组织进行 CMMI 过程改进有所帮助。

参 考 文 献

1 周伯生，吴超英，任爱华，等译. CMMI 精粹－集成化过程改进实用导论. 机械工业出版社，2002

2 CMMI Product Team. CMMI for Development，Version 1.2. CMU/SEI-2006-TR-008. http://www.sei.cmu.edu/cmmi/

3 CMMI Product Team. CMMI Version 1.2 Model Changes. http://www.sei.cmu.edu/cmmi/