概率 XML 文档 Top-k 关键字并行检索算法

周小平 史一民 张 俊
(大连海事大学信息科学技术学院 大连 116026)

摘 要 概率 XML 是描述不确定数据的有效方式,Dewey 编码是一种重要的 XML 文档关键字索引编码技术。在概率 XML 大文档关键字索引检索过程中,频繁地比较关键字索引 Dewey 编码非常耗时。针对上述问题,对概率 XML 文档进行分区,并设计了适合概率 XML 文档特点的关键字索引的 Dewey 编码策略,提出了一种概率 XML 文档 Top-k 关键字并行检索算法 PTKS(Parallel Top-k Keyword Search Algorithm)。实验证明,PTKS 提高了概率 XML 文档关键字检索的时间效率,尤其在文档结构复杂度高的情况下检索效率提高更加显著。

关键词 概率 XML,最小最低公共祖先,XML 文档分区,Dewey 编码,并行检索
中图法分类号 TP311.132 **文献标识码** A

Parallel Top-k Keyword Search Algorithm in Probabilistic XML Documents

ZHOU Xiao-ping SHI Yi-min ZHANG Jun
(School of Information Science and Technology, Dalian Maritime University, Dalian 116026, China)

Abstract Probabilistic XML can describe the uncertain data effectively, and Dewey code is the most important encoding method for indexing probabilistic XML documents. But during the keyword search in big probabilistic XML documents, it takes much time for comparing the Dewey code of keyword index frequently. To deal with the problem above, the probabilistic XML document was partitioned into several fragments, and a new Dewey encoding method of keyword index for probabilistic XML documents was designed, thus, a Parallel Top-k Keyword Search Algorithm (PTKS) was proposed. The experiment results show that the PTKS algorithm has low time complexity, especially, its efficiency is improved significantly when the structure of the document is complicated.

Keywords Probabilistic XML, SLCA, XML document partition, Dewey code, Parallel search

1 引言

近10年来,XML 技术得到了快速发展,针对 XML 数据的检索技术日益成熟。XML 关键字检索技术是当前 XML 检索的主要研究方向,它能够较好地满足用户检索需求,用户不必了解复杂的 DTD 以及 XML 模式,提交关键字检索请求便可得到所需信息。

在信息数据获取过程中,如科研实验、医学研究和污染测试等一系列测试结果都会存在一些不确定信息[1]以及不可避免的误差错误,而这些不确定信息的描述是关系数据库无法给出的。由于 XML 语言能够灵活地描述各种半结构化数据以及非结构化数据,概率 XML 开始成为不确定数据的描述语言,各学科领域以及商业领域已经开始采用概率 XML 描述不确定半结构化数据。

随着概率 XML 数据存储量以及 XML 结构复杂度增大,概率 XML 文档关键字检索出现以下问题:大 XML 文档中,关键字索引的 Dewey 编码存储比较浪费存储空间,频繁地逐段比较 Dewey 编码比较耗时,导致关键字检索时间效率降低。

以往互联网搜索引擎检索结果为单个文本文档或者某个文档链接,而 XML 关键字检索系统中,XML 文档被描述成文档树,检索结果为包含所有关键字的子树,称为最低公共祖先结点(LCA, Lowest Common Ancestor),它促使检索结果粒度更小,更精确。基于 LCA 表示模型,YU 提出了一种新的检索结果模型,称为最小最低公共祖先(SLCA, Smallest Lowest Common Ancestor)[2],它促进了关键字检索结果模型理论的进一步发展。

本文提出了概率 XML 文档 Top-k 关键字并行检索算法 PTKS,其对概率 XML 关键字索引实现分区管理,在概率 XML 文档分区中并行检索关键字信息,以 SLCA 作为检索结果模型,提高了概率 XML 文档关键字检索的时间效率。

2 相关工作及概念定义

2.1 相关工作

(1)Top-k 关键字检索:关键字检索已在关系数据库检索中取得一定研究成果。一个 XML 文档可以看成是一棵有序的、带标签的树,其中元素或属性映射为树结点,元素间嵌套

到稿日期:2012-05-18 返修日期:2012-08-10 本文受国家自然科学基金项目(61073057,60972090)和中央高校基本科研业务费专项资金项目(2011JC007)资助。

周小平(1987—),男,硕士,CCF 学生会员,主要研究方向为 XML 信息检索,E-mail:zxp163@163.com;史一民(1966—),女,硕士,副教授,CCF 高级会员,主要研究方向为智能信息处理;张 俊(1971—),男,博士,副教授,主要研究方向为数据库与信息检索、智能信息处理。

关系映射为边。LCA 是最早提出的 XML 关键字检索结果表示模型，以此为基础，XML 关键字检索技术得到快速发展，涌现出诸多关键字检索算法，如 XSEarch[3]、SLCA、XRank[4] 和 Schema-Free Xquery[5]。其中，XSEarch 是一种由普通 XML 文档的结构化查询方法向关键字检索方法的过渡，为早期 XML 文档的关键字检索算法提供了原始的思想。而 SLCA 继承了 XSEarch 的基本思想，并提出了以栈为辅助工具的关键字检索算法，为 XML 文档关键字检索奠定了基础。XRank 和 Schema- Free Xquery 分别实现了基于栈结构的关键字检索算法，将包含所有关键字的结果集合返回给用户。Li 等人提出了一种新的检索结果表示模型 VLCA（Valuable LCA）[6]，用以避免出现错误肯定问题和错误否定问题。阈值检索[7]是当前 Top-k 关键字检索的主要方法，它通过阈值保留符合条件的 k 个最优结果，实现 XML 文档的 Top-k 关键字检索。

(2)概率 XML：近几年，概率 XML 理论的研究已取得较大成果，Nierman 首先提出了包含独立分布和互斥分布类型的概率 XML 模型 ProTDB[8]。Kimelfeld[9] 对以往概率 XML 研究工作进行总结，肯定了 Nierman 的概率 XML 模型的意义，对概率 XML 模型做了进一步扩展，并对分布结点类型的数据描述能力进行了分析。Chen[10] 综述了概率 XML 结构化查询 Top-k 检索的思想，并提出了相应的概率 XML 文档 Top-k 关键字查询算法。Li[11] 也讨论了在概率 XML 文档下关键字检索处理的方法，并给出了相应的剪枝策略的关键字检索算法。

2.2 相关概念定义

本文采用文档树模型表示 XML 文档，在树模型 T 中，$v\prec v'$ 表示结点 v 是结点 v' 的祖先结点，$v\prec=v'$ 表示结点 v 是结点 v' 的祖先结点或者 v' 本身。

定义 1(LCA[2]) 给定 m 个结点 $v_1,v_2,\cdots,v_m\in T$，称结点 $w\in T$ 为 m 个结点的最低公共祖先(LCA)，当且仅当满足下列条件：

(1)对于 $\forall v_i(1\leqslant i\leqslant m)$ 都满足 $w\prec v_i$；

(2)在 T 中不存在结点 w'，对于 $v_i(1\leqslant i\leqslant m)$ 都满足 $w'\prec v_i$，并且 $w\prec w'$；

将 $v_1,v_2,\cdots,v_m$ 的最低公共祖先 w 记为 $LCA(v_1,v_2,\cdots,v_m)$。

定义 2(SLCA[2]) 文档树 T 中，给定关键词的列表 $\{w_1,w_2,\cdots,w_m\}$，S_i 表示包含关键词 w_i 的结点集。结点 $v_i\in S_i(1\leqslant i\leqslant m)$，表示包含关键词 w_i 的结点。称 v 为结点集合 $S_1,S_2,\cdots,S_m$ 的最小最低公共祖先(SLCA)，当且仅当 v 满足下列条件：

(1)存在结点 v，使得 $v_i\in S_i(1\leqslant i\leqslant m)$，$v=LCA(v_1,v_2,\cdots,v_m)$，记 $v\in LCA(S_1,S_2,\cdots,S_m)$；

(2)在 T 中不存在结点 $v'\in LCA(S_1,S_2,\cdots,S_m)$，使得 $v\prec v'$，将 v 记为结点 $v_1,v_2,\cdots,v_m$ 的最小最低公共祖先，$v\in SLCA(S_1,S_2,\cdots,S_m)$。

定义 3(概率 XML 模型[8]) 在概率 XML 文档中，元素结点分为普通结点和分布结点两种。其中，分布结点又包含 IND 和 MUX 两种类型的元素，其表示形式为 PrXML$^{\{IND,MUX\}}$。IND 类型表示该结点下的孩子结点概率相互独立存在；MUX 类型表示该结点下的孩子结点概率互斥存在，即该结点下至多只有一个孩子结点作为结果存在。一个表示国家信息的概率 XML 的树模型如图 1 所示。

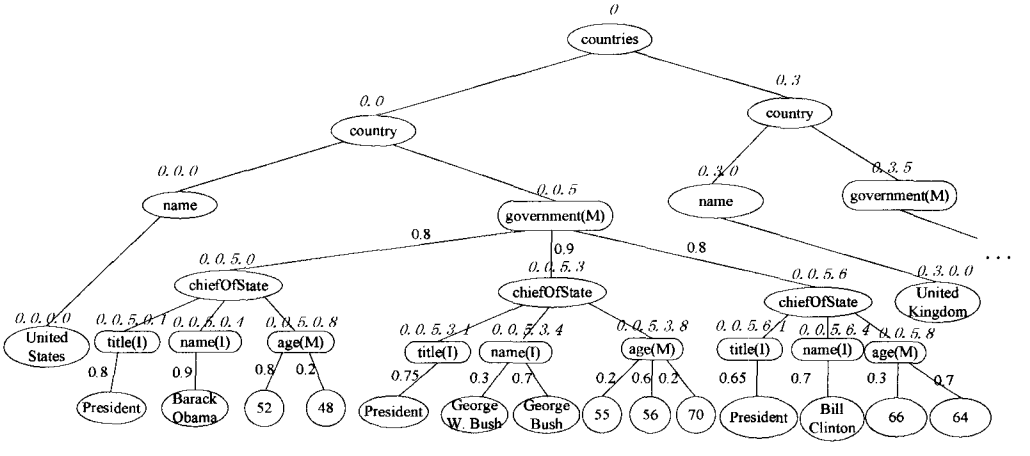

图 1 概率 XML 树模型[12]

其中，椭圆代表普通的 XML 元素结点，而圆角矩形表示分布结点。

定义 4(概率 XML 文档可能世界[11]) 概率 XML 文档中，根据定义 3 中的分布结点的类型，所有分布结点的孩子结点有多种组合情况，每个以该结点为根的孩子结点的组合与该结点的父亲结点所在的 XML 树构成了概率 XML 文档可能世界的一个实例。则消除概率 XML 文档树中的所有分布结点所形成的所有概率 XML 文档实例的集合称为概率 XML 文档可能世界，并用 p-document 表示。

定义 5(概率 XML 文档 Dewey 编码策略) 给定带标签有向树 $T=(V_T,E_T,r,A,type)$，其中，V_T 表示树中结点的集合；E_T 表示树中所有边的集合；r 为有向树的根；A 表示所有结点标签的集合；$type$ 为 T 中元素结点类型，包括 $\{O,I,M\}$，其中 O 表示普通结点类型，I 表示独立分布类型，M 表示互斥分布类型。T 中任意结点的 Dewey 编码生成分为两步，首先，按照式(1)根据当前结点的兄弟结点的当前 ID 编码确定该结点的当前 ID 编码 CurID，它包含了当前结点的类型和序的信息；其次，按照式(2)合并 CurID 和该结点的父亲结点的 Dewey 编码，形成当前结点的 Dewey 编码。具体的编码规则执行方式如式(1)、式(2)所示。

(1) 给定结点 $v_i(i\geqslant 0)$，i 表示结点 v 在兄弟结点间的位置，则结点 v_i 的当前 ID 编码方法如式(1)所示。

$$\mathrm{CurID}(v_i)=\begin{cases}0, & \text{if } i=0 \text{ and } type(v_i)=O\\ 1, & \text{if } i=0 \text{ and } type(v_i)=I\\ 2, & \text{if } i=0 \text{ and } type(v_i)=M\\ \lceil CurID(v_{i-1})/3\rceil *3, & \text{if } type(v_i)=O \text{ and } CurID(v_{i-1})\%3\neq 0 \text{ and } i>0\\ \lceil CurID(v_{i-1})/3\rceil *3+3, & \text{if } type(v_i)=O \text{ and } CurID(v_{i-1})\%3=0 \text{ and } i>0\\ \lceil CurID(v_{i-1})/3\rceil *3+1, & \text{if } type(v_i)=I \text{ and } CurID(v_{i-1})\%3\neq 0 \text{ and } i>0\\ \lceil CurID(v_{i-1})/3\rceil *3+4, & \text{if } type(v_i)=I \text{ and } CurID(v_{i-1})\%3=0 \text{ and } i>0\\ \lceil CurID(v_{i-1})/3\rceil *3+2, & \text{if } type(v_i)=M \text{ and } CurID(v_{i-1})\%3\neq 0 \text{ and } i>0\\ \lceil CurID(v_{i-1})/3\rceil *3+5, & \text{if } type(v_i)=M \text{ and } CurID(v_{i-1})\%3=0 \text{ and } i>0\end{cases} \tag{1}$$

(2) 给定结点 u,v，若 u 是 v 的父亲结点，根据结点 u 的 Dewey 编码可以方便地得到结点 v 的 Dewey 编码，如式(2)所示：

$$Dewey(v)=\begin{cases}0, & \text{if } v=\text{root}\\ Dewey(u)+\text{"."}+CurID(v), & \text{if } v\neq \text{root}\end{cases} \tag{2}$$

概率 XML 文档树中各元素结点的 Dewey 编码如图 1 所示，图中包含 4 个分布结点：{government，title，name，age}，其中{title，name}为独立分布结点类型，{government，age}为互斥分布结点类型。图 1 中 title 结点的 Dewey 编码为{0.0.5.0.1，0.0.5.3.1，0.0.5.6.1}。

概率 XML 文档的 Dewey 编码策略是构建概率 XML 文档关键字索引的核心内容。在进行编码的同时，记录当前结点所包含的关键字信息，为每个关键字构建倒排索引，具体的表示形式为：$Index_k=\langle Parid, Dewey, Prpath\rangle$，其中，$Parid$ 表示元素结点的分区 ID，$Dewey$ 表示元素结点在概率 XML 分区内的编码，$Prpath$ 表示元素结点的路径概率分布表。

3 概率 XML 文档 Top-k 关键字并行检索系统

3.1 系统结构

本文系统主要由 3 个模块组成，即文档分区处理模块、关键字索引构建模块以及关键字检索处理模块。系统结构如图 2 所示。

首先，通过文档解析器解析概率 XML 文档，文档分区处理器对概率 XML 文档进行分区，将各分区记录信息保存在分区索引库中，关键字索引构建处理器将各文档分区内包含的关键字形成倒排索引。当用户提交检索请求时，关键字检索控制器(SC，Search Controller)分析用户输入得到关键字列表，并向每个分区关键字检索处理器(SP，Search Processor)发出关键字检索命令。系统检索过程中，各分区内关键字索引的处理并行执行，各关键字索引处理器根据 SC 发送的检索命令，提取各分区内包含关键字的候选结点集，并将匹配所有关键字的 SCLA 结点发送到 SLCA 概率计算处理器，计算其概率，将概率最大的前 k 个 SLCA 结点返回给用户。

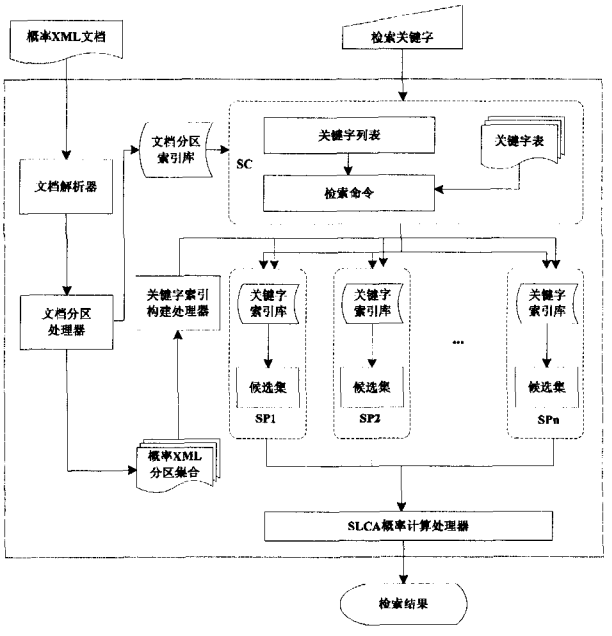

图 2 概率 XML 文档 Top-k 关键字并行检索系统

3.2 概率 XML 文档分区

为支持大数据量 XML 文档中高效关键字的检索，Kurita[13]，Zhang[14] 分别提出在分布式环境下对 XML 文档进行数据分区以及并行检索的思想，对 XML 文档实现分区存储及相关检索，以满足用户检索信息的时间要求。

本文将 Zhang 的分区思想应用在静态概率 XML 文档上。从概率 XML 树形结构以及文档内容两个角度，选取合适的层次深度和合理的索引元素粒度，初始化树形索引集合，对创建的 XML 树索引进行分区。每个分区索引包含其祖先路径信息以及当前 ID 信息，其表示形式为：

$$SubPIndex=\{subtree\_id, current\_id, flag, prefix\}$$

其中，$subtree\_id$ 表示子树在分区中的 $Dewey$ 编码；$current\_id$ 表示元素所在 XML 的分区 ID；$flag$ 表示元素结点是否为 SLCA 结点；$prefix$ 表示该元素结点的路径编码信息。

设原始 XML 文档大小为 S，C 为需要分区的数量，则理想的分区大小为 $M=S/C$。为保证文档分区结构的完整性，需要为文档分区大小设定一个合理的区间$[M(1-\varepsilon), M(1+\varepsilon)]$，其中 ε 为大于零的较小的值。

如果当前子树规模 $sc\in\Omega$，则需要将当前子树划分出去作为一个分区；如果当前子树规模 $sc<M(1-\varepsilon)$，则合并该子树的兄弟子树直至规模 $sc\in\Omega$，并将这些兄弟节点构成的子树划分出去作为一个分区，再将当前子树结点的前缀码信息加入到 $recordSet$ 中，继续执行其它子树分区操作；如果当前子树规模 $sc>\theta(1+\varepsilon)$，则将结点 n 的编码信息加入到 $prefix$ 中，对其子树递归执行规模统计，将符合条件的概率 XML 子树分割出去作为一个单独的分区，直至所有子树均被分配完毕。本文的概率 XML 分区算法(Probabilistic XML Partition Algorithm，PXPA)如算法 1 所示。

算法 1 PXPA 算法

```
输入:T 的根结点 n;结点 n 的 Dewey 编码 prefix;
输出:分区记录信息 recordSet
1. subtree_id=get_id(n);
2. current_id=get_currentid();
3. if (sizeof(n)<=θ(1+ε) && sizeof(n) >=θ(1+ε))
4. content=get_content(n);
5. recordSet. addrecord(subtree_id,currentid,content);
6. else if (sizeof(n)<=θ(1-ε))
7.      content +=get_content(n);
8. else
9. pre_id=id+n. local_id;
10. p_path=p_path +n. name;
11. List eleList=n. nodes();
```

3.3 可能世界构建

由于概率 XML 文档中存在分布结点，用传统的 SLCA 检索算法进行概率 XML 的关键字检索存在错误肯定问题和错误否定问题，因此传统的 SLCA 检索算法已无法满足概率 XML 的检索要求。一般解决方式是将概率 XML 分区子树转换成普通 XML 子树，构建概率 XML 文档的可能世界，并依次对每个普通 XML 文档执行关键字检索算法。

概率 XML 文档可能世界的构建过程就是将分布类型结点转换成多个普通 XML 子树的过程。本文概率 XML 树分布结点的转换处理分为两部分，即 IND 结点处理和 MUX 结点处理。

(1) IND：对于 IND 元素结点，假设结点 T' 包含 n 个孩子结点，由于其孩子结点存在概率相互独立，因此该元素结点的孩子结点存在的可能情况有 2^n 种。这里，用长度为 n 的二进制 Dewey 编码以及 IND 结点类型对孩子结点进行标识，生成 2^n 个该文档树的副本，删除当前分布结点并将每种孩子结点的组合添加到 T' 的父结点中。

(2) MUX：对于 MUX 元素结点，假定结点 T' 包含 n 个孩子结点，由于其孩子结点存在互斥，该概率结点孩子结点的存在有 $(n+1)$ 种组合方式。这里用长度为 n 的二进制 Dewey 编码以及 MUX 结点类型对孩子结点进行标识，生成 $(n+1)$ 个该文档树的副本，删除当前分布结点并将每种孩子结点的组合添加到 T' 的父结点中。

3.4 SLCA 概率计算

概率 XML 文档转换成概率 XML 可能世界 XML 树集合，则概率 XML 文档的 Top-k 关键字检索转换成在概率 XML 可能世界 XML 树集合中检索 SLCA。

给定一组关键字 $\{w_1, w_2, \cdots, w_n\}$ 以及整数 k，则概率 XML 文档的 Top-k 关键字检索可以理解为在概率 XML 文档中计算概率最大的 k 个 SLCA 结点。因此，SLCA 概率计算方法为：列举概率 XML 在可能世界中产生的多个子树，依次计算每个子树中该结点成为 SLCA 的概率并求和。保留概率最大的前 k 个 SLCA 结点，其计算公式为：

$$\Pr^G_{slca}(v)=\Pr(path_{r\to v})\times \Pr^L_{slca}(v) \tag{3}$$

$$\Pr^L_{slca}(v)=\sum_{i=1}^{m'}\{\Pr(t_i)\mid slca(v,t_i)=\text{true}\} \tag{4}$$

式中，$\Pr(path_{r\to v})$ 表示结点 v 在概率 XML 分区子树中存在的概率，$\Pr^L_{slca}(v)$ 表示在以 v 为根结点的树转换成的各普通 XML 子树中 v 为 SLCA 的概率。m' 表示 v 产生的子树个数，t_i 为 v 产生的某一子树 $(1\leqslant i\leqslant m')$，$slca(v,t_i)=\text{true}$ 表示 t_i 存在 SLCA 结点 v。

则在所有文档分区中，检索结果 SLCA 元素结点集合表示为：

$$ElemSet(R_{slca})=\{v_i\mid prob(v_i)>\varepsilon, 0<i<k+1, v_i\in R_{slca}\} \tag{5}$$

式中，R_{slca} 为所有分区下 SLCA 结点构成的集合，v_i 为 R_{slca} 集合中，$\Pr^G_{slca}(v)$ 为概率最大的前 k 个元素结点。

本文用概率分布表表示概率 XML 中的每个元素结点关键字分布情况，二进制编码表示关键字的包含情况。给定 n 个关键字 $Q=\{w_1, w_2, \cdots, w_n\}$，结点 p 的局部概率为 λ_i，包含 m 个孩子结点，则结点 p 的概率分布表 tab_p 为包含 2^n 个项的一维表，其各项值由其孩子结点依次合并累加得到。其概率计算方式如式(6)—式(9)所示。

1)若结点 p 类型为 IND，其孩子结点 C_i $(1\leqslant i\leqslant m)$ 的概率分布表为 $tab_i=\{\mu_{(2^n-1)}, \cdots, \mu_0\}$，则将 C_i 的局部概率与其概率分布表相乘，并按照式(6)以及式(7)处理。

$$tab_i^{\uparrow}=\begin{cases}\{x\mid u_x=\lambda_i * u_x, \forall x\in tab_i, x\neq 0\}\\ \{x\mid u_x=1-\lambda_i * (1-u_x), x=0\}\end{cases} \tag{6}$$

结点 p 的概率分布表 $tab_p=\{\mu_{(2^n-1)^p}, \cdots, \mu_{0^p}\}$，将 p 的孩子结点的概率分布表各项相与并累加得到：

$$tab_{p'}=\{z\mid \mu_z=\sum \mu_x * \mu_y, z=x\text{OR}y, \forall x\in tab_i^{\uparrow}, \forall y\in tab_p\} \tag{7}$$

若 p 的概率分布表 tab_p 初始为零，则将 C_i 的概率分布表直接赋值到 tab_p 中即可，然后依次合并处理 p 的孩子结点。

2)若结点 p 类型为 MUX，同样将 p 的孩子结点 C_i 的概率分布表各项与其局部概率相乘，如式(8)所示。

$$tab_i^{\uparrow}=\{x\mid \mu_x=\lambda_x * \mu_x, \forall x\in tab_i\} \tag{8}$$

因此，可以将计算得到的 $tab_i^{\uparrow}$ 直接依次按序加到 tab_p 中，得到更新后的概率分布表 $tab_{p'}$，如式(9)所示。

$$tab_{p'}=\{z\mid \mu_z=\mu_x+\mu_z, z=x, \forall x\in tab_i^{\uparrow}, \forall z\in tab_p\} \tag{9}$$

3)若 p 结点为普通结点，其计算方式与式(5)中一样，区别在于普通结点可以产生 SLCA，所以当 tab_p 的“111…11”项的值不为零时，把结点 p 作为 SLCA 结果返回，并作进一步处理，同时将“111…11”项的概率值置零，继续将 p 的概率分布向上合并操作，直至根结点。

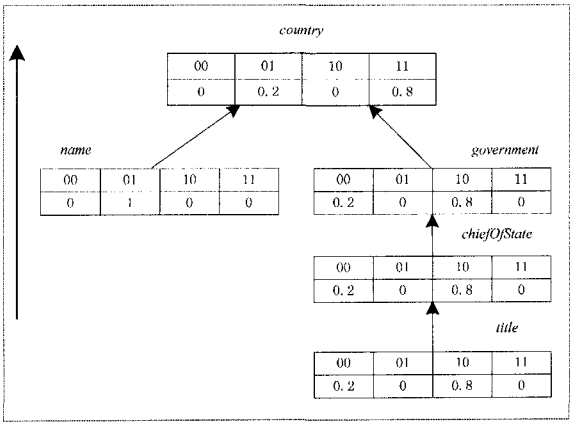

图 3 SLCA 概率计算过程

以图1所示概率XML树为例，给定关键字{“United States”，“President”}，提取包含关键字的候选结点{0.0.0.0，0.0.5.0.1，0.0.5.3.1，0.0.5.6.1}，初始化候选结点概率分布表。按照自底向上的计算方式，首先合并结点name(0.0.0.0)和结点title(0.0.5.0.1)的概率分布表，将概率分布表各项与其局部概率相乘，得到中间概率分布表；按照式(6)－式(9)依次对不同类型的结点执行不同的概率分布表合并操作，得到结点country的概率分布表，可以看出“11”项概率不为零，即Pr^{G}_{slca}(country)＝0.8，证明该结点为SLCA结点，将该结点的概率信息及内容信息加入到结果集中。同理可以计算出其它两个结点的SLCA，概率分别为0.75，0.65，假设k＝2，则结点name(0.0.0.0)和title(0.0.5.0.1)的SLCA以及name(0.0.0.0)和title(0.0.5.3.1)的SLCA为概率最大的两个结果。name(0.0.0.0)和title(0.0.5.0.1)的SLCA计算过程如图3所示。

3.5 PTKS算法

该算法需要用到两个辅助数据结构：其一，哈希映射表，表示结点与关键字的分布的映射关系，记为φ:v→$dist$，结点采用Dewey编码标识，关键字分布情况采用dist表示。其二，路径概率表，表示从根结点到其父结点之间所有结点的概率，记为γ:v→$Prpath$。

给定一组关键字Q={w_1，w_2，…，w_n}以及概率XML文档分区索引SubPIndex。首先，根据倒排索引提取关键字候选表中的编码最小的关键字结点v，初始化结点v的Dewey编码，并依次处理其相邻结点。按照式(3)－式(9)对结点进行处理，依次将中间结点p的概率与其孩子结点的概率分布表进行合并，若p为分布结点，按照不同的结点类型处理方式，执行n次合并便可得到结点p的概率分布表，它的每项概率值表示结点p映射的概率XML文档可能世界的实例所包含关键字的概率和。在合并概率分布表的过程中，提取包含所有关键字的SLCA结点相关信息，按照迭代的方式自底向上地依次计算其概率，直至所有包含关键字的候选结点被处理完毕，算法执行结束。PTKS算法执行结果为概率最大的前k个SLCA结点集合。算法PTKS如算法2所示。

算法2 PTKS算法

输入：关键字Q{w_1，w_2，…，w_n}；概率XML文档分区索引SubPIndex。

输出：概率最大的k个SLCA结点构成的集合R。

```
1. 加载包含关键字的候选结点以及路径概率；
2. 取最小编码结点 v 为第一个处理结点；对 v 的路径编码进行初始化；
3. While L. Next()!= null
4. { v'=GetNextElem(L);
5.   if(! isPromoted(v'))
6.   { //求 v 和 v'的最大公共路径。
7.     pre=lcp(stack,v');
8.     while(stack. size>p. length)
9.     { v=stack. pop();
10.      type=gettype(v);
11.      prob= lcpprob(v);
12.      if(type=='o')
13.      { Lock(recordSet);
14.        dist=map. Getdist(v);
15.        if (isSLCA(v))
16.        { prob=GetlcpProb(v)
17.          path_prob=getpathprob(v. Prpath);
18.          prob_all= prob * path_prob;
19.          if(R. length < k) {R. push(v);}
20.          else if (prob_all>prob(R_k)) { R. pop(R_k);R. push(v);}
21.          v. dist("11…11"). prob=0;}
22.        map. update(v. dist);
23.        Unlock(recordSet);}
24.      if(map. contains(p(v)))
25.        UpdateDist(v,type,prob,p(v));
26.      else
27.        CombineProb(v,type,prob,p(v));}
28.    for j=pre. length to v'. length stack. push(v'[j]);
29. }
30. return R;
```

4 实验分析

系统的硬件平台为3台配备Intel Pentium(R) 4、主频2.99 GHz CPU、2G内存的PC机，软件平台为Java并行处理框架(JPPF，Java Parallel Processing Framework)，其用来实现概率XML文档的并行检索。JPPF是一个开放源码的网格计算框架，它支持在分布式环境中的并行计算。本实验采用DBLP[15]、SwissPro[16]以及NASA[17]作为测试数据集，由于原始的DBLP、SwissPro以及NASA数据集中结点不包含概率分布信息，本文所用数据集是根据文献[9]中的方法对DBLP、SwissPro和NASA进行转换，得到概率XML数据集DBLP_P、SwissPro_P和NASA_P。DBLP_P数据量大，文档结构相对简单；NASA_P文档XML树结构层次深，文档结构则相对复杂；而SwisPro数据集复杂度则居于两者之间。实验比较了传统关键字索引构建方法与PXPA方法所构建的概率XML关键字索引的存储开销，并依次比较了不同分区粒度下关键字检索算法的时间效率。

4.1 PXPA算法评价

两个文档的规模信息如表1所列。将三分区下执行PXPA算法后生成的关键字索引数据存储量与文献[17]的单分区概率XML文档元素的索引编码数据存储量相比较，结果如表2所列。其中，DeweyXML表示未对原始概率XML文档作分区处理，保持原文档结构不变。由于概率XML文档分区子树的祖先结点路径保存在分区索引库中，即表2中的辅助存储空间，因此概率XML文档树中的元素结点总数未改变。

表1 概率XML文档

数据集	文件大小(MB)	文档深度	平均深度	结点数
DBLP_P	22.74	6	2.90	1069561
SwissPro_P	21.63	5	3.6	928465
NASA_P	23.37	8	5.58	1054559

表2 DeweyXML与PXPA XML元素索引存储比较

算法	辅助存储(MB)	索引存储(MB)		
		DBLP_P	SwissPro_P	NASA_P
DeweyXML	无	16.36	15.68	19.42
PXPA	1	9.24	9.34	10.75

由表1、表2结果可以看出，本文提出的PXPA算法与传

统的单分区 XML 文档索引生成算法相比，其关键字索引占用的存储空间更小，减少了关键字检索过程中系统内存的开销。

4.2 PTKS 算法评价

本文从不同分区数量的角度分别采用 PTKS 算法对 DBLP_P、SwissPro_P 以及 NASA_P 3 个数据集的关键字检索时间进行比较，如图 4－图 6 所示，其中，检索结果数量设定为 20。在图 4－图 6 的基础上，比较了不同数据集上关键字个数相同的情况下，单分区与三分区检索时间的比值，如图 7 所示。

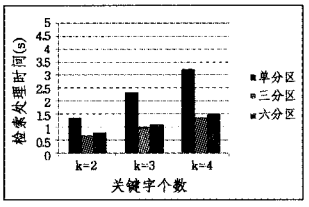

图 4　不同分区下 DBLP_P 数据集检索时间

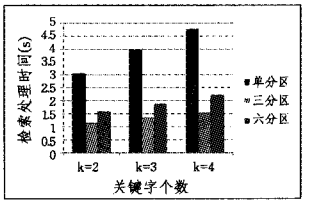

图 5　不同分区下 SwissPro_P 数据集检索时间

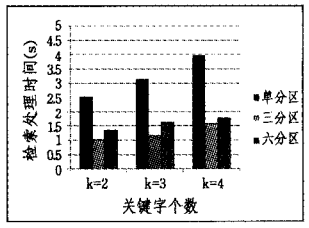

图 6　不同分区下 NASA_P 数据集检索时间

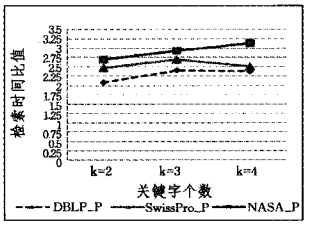

图 7　单分区与三分区检索时间比值

由图 4－图 6 可以看出，与未作分区处理的单分区文档检索时间相比，PTKS 算法的检索时间整体显著降低，但分区数量不同，算法的检索处理时间就不同，并不是分区数量越多其处理时间就越少，因此，确定合适的文档分区粒度对算法的时间效率有一定的影响。同时，随着关键字个数的增加，PTKS 算法的检索处理时间也会显著增加。

在图 7 中，关键字个数相同情况下，复杂度相对较高的 SwissPro_P 和 NASA_P 数据集，其单分区下的检索时间与三分区下的检索时间比值更大。由此可以看出，PTKS 算法对于复杂度较高的数据集，在检索时间效率的提高上更加显著。

图 8 展示了在三分区、关键字个数为 3 的情况下，检索结果数量 Top-k 对 PTKS 算法的时间效率的影响。

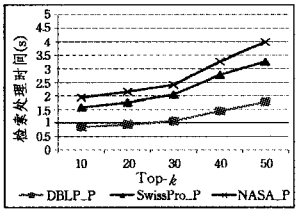

图 8　三分区关键字 Top-k 检索时间

从图 8 可以直接看出，随着 Top-k 值的增大，PTKS 算法的检索时间也相应增大，关键字检索时间效率降低。随着 Top-k 值的增大，概率计算处理器与各分区处理程序的通信量增加，SLCA 结点在结果集内的排序时间也相应增加，导致 PTKS 算法的时间效率降低。

结束语　概率 XML 文档分区是实现概率 XML 文档关键字并行检索的基础。本文将分布式环境下普通 XML 文档分区思想引入到静态概率 XML 文档中，提出了概率 XML 文档 Top-k 关键字并行检索算法 PTKS，解决了大 XML 文档中关键字检索效率低的问题。实验验证了多分区下 PTKS 算法的高效性，对于复杂度高的概率 XML 文档，关键字检索的时间效率提高更明显。PTKS 算法忽略了概率 XML 文档中存在的引用关系，该情况下的概率 XML 文档关键字检索还有待于进一步探讨。

参 考 文 献

[1] 周傲英，金澈清，王国仁，等. 不确定性数据管理技术研究综述[J]. 计算机学报，2009，32(1)：1-16

[2] Xu Yu，Papakonstantinou Y. Efficient Keyword Search for Smallest LCAs in XML Databases [C]//SIGMOD. 2005：537- 538

[3] Cohen S，Mamou J，Kanza Y，et al. XSEarch：A Semantic Search Engine for XML[C]//VLDB. 2003：45-56

[4] Guo Lin，Shao Feng，Botev C，et al. XRANK：Ranked Keyword Search over XML Documents[C]//SIGMOD. 2003：16-27

[5] Li Yun-yao，Yu Cong，Jagadish H V，et al. Schema-Free XQuery [C]//VLDB. 2004：72-83

[6] Li Guo-liang，Feng Jian-hua，Wang Jian-yong，et al. Effective-Keyword Search for valuable LCAs over xml documents [C]//CIKM. 2007：31-40

[7] Fagin R，Lotem A，Naor M，et al. Optimal aggregation algorithms for middleware [J]. Comput. Syst. Sci，2003，66(4)：614-656

[8] Nierman A，Jagadish H V. ProTDB：Probabilistic Data in XML [C]//VLDB. 2002：646-657

[9] Kimelfeld B，Sagiv Y. Modeling and querying probabilistic XML data[J]. SIGMOD Record (SIGMOD)，2008，37 (4)：69-77

[10] Chen L J，Papakonstantinou Y. Supporting Top-K Keyword Search in XML Databases[C]//ICDE. 2010：689-700

[11] Li Jian-xin，Liu Cheng-fei，Zhou Rui，et al. Top-k Keyword Search over Probabilistic XML Data [C]//ICDE. 2011：673-684

[12] Ning Bo，Liu Cheng-fei，et al. Matching Top-k Answers of Twig Patterns in Probabilistic XML[C]//DASFAA. 2010：125-139

[13] Kurita H，Hatano K，Miyazaki J，et al. Efficient Query Processing for Large XML Data in Distributed Environments [C]//AINA. 2007：317-322

[14] Zhang Chen-jing，Ma Qiang，Wang Xiao-ling，et al. Distributed SLCA- Based XML Keyword Search by Map-Reduce [C]//DASFAA Workshops. 2010：386-397

[15] DBLP [DB]. http://dblp. uni-trier. de/xml，2012-03-31

[16] SwissPro [DB]. http://www. cs. washington. edu/search/xml-datasets /www/ repository. html# SwissPro，2001

[17] NASA [DB]. http://www. cs. washington. edu/search/xmldatasets/www/ repository. html#NASA，2001

[18] 吴海涛，唐振民. XML 文档的 Dewey 编码生成算法[J]. 计算机工程，2011，36(19)：62-64