MXDR:一种基于关键字的XML多文档分布式检索方法

李 霞 李战怀 张利军 陈 群 李 宁
(西北工业大学计算机学院 西安 710072)

摘 要 基于关键字的XML检索技术是近几年信息检索领域的研究热点。但是由于关键字缺少XML结构语义信息,检索结果和用户需求偏差较大,检索质量难以提高;而XML结构检索由于用户难以提出准确描述查询意图的查询表达式而难以普及。另一个更突出的问题是现有的XML检索研究绝大多数都集中在单文档上,缺乏实用性。因此提出一种基于关键字的结构检索方法,即用分布式方式实现对多XML文档的检索,简称为MXDR(Multi-XML Distributed Retrieval)。MXDR首先用一种兼顾结构和内容的聚类方法对多文档进行分类,通过分析查询关键字和类别结构信息,确定分布查找策略,再结合查询关键字和XML的结构信息,构建结构查询语句,最后通过结构查询系统实现关键字检索。在多组真实数据Sigmod数据集上的验证结果表明,与经典的SLCA方法比较,MXDR方法具有较高的查全率和查准率,尤其在检索效率上MXDR方法有显著优势。

关键词 XML多文档,关键字检索,结构检索,分布式
中图法分类号 TP311 **文献标识码** A

MXDR:Distributed Information Retrieval for Multi-XML Document Based on Keywords

LI Xia LI Zhan-huai ZHANG Li-jun CHEN Qun LI Ning
(School of Computer Science and Technology, Northwestern Polytechnical University, Xi'an 710072, China)

Abstract The emergence of the Web has increased interests in XML data. Keyword search has attracted a great deal of attention for retrieving XML data because it is a user-friendly mechanism. But Keyword search is hard to directly improve search quality because lots of keyword-matched nodes may not contribute to the results. A more important issue is the current studies are focused on single XML retrieval, lack of practicability. To address the challenge, this article proposed a new approach for automatically correcting queries over Multi-XML, called MXDR(Multi-XML Distributed Retrieval). We first classed multi-XML documents by a clustering method, and elicited the common structure information. Then generated certifiable structured queries by analyzing the given keywords query and the common structure information of XML datasets. We can evaluate the generated structured queries over the XML data sources with any existing structure search engine. We conducted an experimental study on real-life multi-XML datasets. The experimental results show that MXDR is effective and efficient in supporting structural queries, compared with existing proposals.

Keywords Multi-ML, Keywords IR, Structure IR, Distributed

1 引言

XML检索技术主要有两个研究方向:基于结构检索和基于关键字检索。基于结构检索的XML检索技术早先得到了广大研究者的关注,已取得了较成熟的成果[1,2]。基于关键字的XML检索技术,由于其用户友好性,近几年得到了信息检索领域研究者们的广泛关注[3-8]。但是XML关键字检索,由于缺少XML的结构语义信息,检索结果往往与用户需求存在较大偏差,而XML结构检索,用户不仅需要了解XML的结构,还需要掌握结构检索语言,尤其是检索异构的XML文档时,更增加了用户编写结构检索语句的不便和困难。该问题在多文档检索时更加突出。如能采用关键字检索方式,由查询系统根据XML数据源的结构语义特征,构建符合用户查询需求的结构查询语句,则既解决了用户难以描述查询需求的问题,又能提高检索质量。笔者就该问题,提出了一种基于关键字的结构查询的解决方案,已撰写成文[9]。

XML信息检索研究现状中,更突出的问题是,大多数研究者仅针对单个XML文档开展研究,显然缺乏实用性。而多文档检索时,若采用传统的单文档检索方式,当数据集规模较大时,检索效率问题随之凸显。分布式检索方案,可通过预处理缩小检索范围,同时由于分布式检索的可同步性,可大大缩短检索时间,提高检索效率。因此,分布式检索是多文档检索的有效解决方案。

到稿日期:2010-12-28 返修日期:2011-03-28 本文受国家自然科学基金(60803043,60970070)和国家高技术研究发展计划(863)(2009AA1Z134)资助。

李 霞(1977-),女,博士生,讲师,主要研究领域为XML信息检索、数据管理等,E-mail:lixia@nwpu.edu.cn;李战怀(1961-),男,教授,博士生导师,主要研究领域为数据库系统、数据管理等;张利军(1978-),男,博士生,主要研究领域为XML数据管理等;陈 群(1976-),男,教授,博士生导师,主要研究领域为XML数据管理、RFID数据管理等;李 宁(1978-),女,博士生,主要研究领域为软件可靠性等。

2 相关工作

有关 XML 关键字检索和结构检索的相关研究，笔者在文献[9]中有详细阐述，本文略。本文侧重 XML 多文档分布式检索的集合划分方法和检索策略。

集合划分是影响分布式信息检索效果的一个重要问题。有效合理的集合划分，将大大缩小检索范围，提高检索效率。理想的集合划分是，一个查询的相关文档集中在一个或少数集合中。如果文档和查询之间是“一对一”或者“多对一”的关系，那么只要将相关文档放入一个子集合即可；但文档和查询之间往往是“多对多”的关系，因此需要研究符合实际情况的划分策略。

Claudine Badue 等人在文献[10]中给出了划分索引的两种策略：第一种是把文档分布在多台处理机上，并行建立索引，采用并行检索方式；另一种是，将建好的倒排索引按照关键词的顺序分布在不同的机器上，每个机器负责处理一部分关键词的倒排索引，在检索时只检索含有查询关键词的机器。第一种策略中，文档的分布是随机的，在检索时采用并行处理办法，对所有文档集合进行检索，因此没有真正降低检索开销。第二种策略仅从文档内容角度对文档集合进行划分，没有考虑 XML 含有丰富语义信息的结构特征，从而影响检索质量。

Yi[11]等人提出了一种用于半结构化文档分类的扩展向量模型，采用嵌套定义的向量描述文档元素，通过概率统计方法进行分类。Denoyer[12]等人提出了利用贝叶斯网络模型进行半结构化文档分类的方法，这些集合划分的方法考虑了文档的结构特性，但是没有考虑查询对文档集合划分的影响。而文档集合划分的目的是为了更好地实现检索，应该是以有利于检索为导向的划分，如果只关注文档而忽略查询显然不合理。

3 预备知识

本文将 XML 文档建模为一棵 XML 树模型[1,13]，记为 $T=(N,E,r)$，其中 N 为节点集合，$N=NE\cup NV$，其中 NV 为叶子节点集合(值节点集合)，NE 为非叶子节点集合(内节点集合)，E 为边集合，且 $E=N\times N$，$r\in N$ 为根节点。E 满足 r，没有父节点；对 $\forall u\in N, u\neq r, u$ 有且仅有一个父节点。

定义标签函数：$label: NE\rightarrow$String，其中函数 label 返回节点 $u\in NE$ 的标签名。本文用 $label(u)$ 表示节点的标签。对 $\forall u,v\in N$，如 u 是 v 的父节点，记 $parent(v)=u$。如 u 是 v 的祖先节点，记 $anc(v)=u$。T 的内节点 u，以 u 为根的子树记为 $T_u'=(N',E',u)$，其中 u 为 T_u' 的根节点，$N'=N\cap\{v|v\in N\wedge anc(v)=u\}$，$E'=E\cap\{N'\times N\}$。图 1 给出 XML 文档树示例。

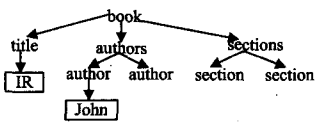

图 1 XML 文档树示例

定义 1(信息对象) 具有独立性、完整性的一个信息实体称为信息对象(Information Object，简称 O)，在 XML 模型树 $T=(N,E,r)$ 中，如果存在一个具有独立完整意义的小枝 T_u'，记为 $T_u'=(N',E',u)$，则记该信息对象为 O_u，对象名为 T_u' 的根节点，记为：$label(O_u)=u$。

定义 2(别名信息对象) 在一棵语义清晰的 XML 文档中，如果存在两个信息对象 O_1 和 O_2，满足以下两个条件：

(1) $label(O_1)=label(O_2)$

(2) $SIM(O_1,O_2)\geqslant\theta$，$\theta$ 为设定的相似度阈值，$\theta\leqslant 1$，则认为信息对象 $O1$ 和 $O2$ 是同一信息对象，互称为别名信息对象(Alias Information Object，简称 AIO)。

定义 3(集合节点) XML 文档树 $T=(N,E,r)$ 中，对 $\forall u\in NE$，如果节点 u 的所有子节点的节点名相同，或所有子节点均互为别名信息对象，则称该节点 u 为集合节点。

4 方法

本文提出一种基于关键字的结构检索方法，用分布式方式实现多 XML 文档检索，简称为 MXDR (Multi-XML Distributed Retrieval)。

MXDR 首先用聚类方法对多文档集合进行划分，该聚类方法综合考虑 XML 结构特征和查询信息，确定文档分布策略，接着用关键字检索的查询方式，通过集合选择，确定查询范围和查询的优先次序，再根据用户输入的关键字，结合 XML 的同类结构信息，构建结构查询语句 XPath，最后用成熟的结构检索系统 Saxon-进行检索。MXDR 分解为以下 3 个关键问题：

1. 如何对 XML 文档集合进行划分；
2. 如何确定分布检索策略；
3. 如何根据用户输入的查询关键词构造结构查询语句。

其中第 3 点，笔者已撰写成文，详见文献[9]，本文略。

4.1 文档集合划分

分布式信息检索过程一般分为：文档集合的划分、检索集合的选择、检索过程的实施。集合划分是构建分布式检索系统的第 1 步，也是分布式信息检索重要的一步。本文提出一种全新的检索方案 MXDR：用分布式实现基于关键字、由关键字构造结构查询语句的检索方法。因此，针对此检索方式，理想的文档集合划分方法是，同结构的文档分到一个集合中，兼顾内容。同结构的文档归为一类，易于提炼集合共通结构信息，便于构造准确的结构查询语句，且能提高检索的效率。在判断结构相似性时，用定义 3(集合节点)的概念，规避用词不统一而导致的同类文档划分为不同类的现象，从而提高集合划分的准确度。

4.1.1 划分方法

一般情况，结构相似的文档语义接近，因此将相似结构的文档划分在同一集合中，同时提取共同结构信息，并推测集合的主题信息，构成集合的特征信息表，供集合选择用。下面说明集合划分的方法。

依次扫描 XML 文档，抽取文档的结构序列，如图 2 所示，从文档 1 中抽取出 4 个结构序列，记入图 2(c)中，文档 2 抽取出 4 个结构序列，记入图 2(d)中。通过定义 2 (别名信息对象)的判断公式，判断集合节点，归并相似结构序列，得到频繁子序列，如图 2(e)所示。将 XML 文档表示为|F|维的向量，每个分量值为 1 或 0，如果该文档包含特征空间中对应的路径，则值为 1，否则为 0。得两个 XML 文档 d_i，d_j 间的相似性定义为：

$$\mathrm{Sim}(d_i,d_j)=m/|F| \tag{1}$$

式中，m 为 d_i 和 d_j 的向量中同时取值为 0 或者同时取值为 1

的属性个数。例如：$Sim(d_1,d_2)=3/4$，如图 2(f)所示。

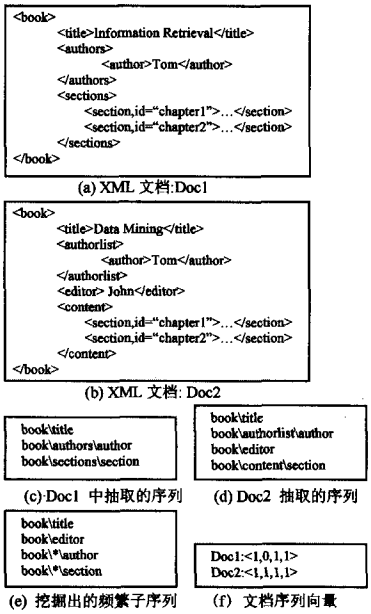

图 2 XML 文档树和结构序列示例

本方法的特点是，在挖掘频繁子序列时，通过对集合节点的判断和别名信息对象的判断，归并了相似结构序列，解决了因用词不统一，误将同结构文档判断为不同结构的问题。例：通过集合节点的判断，合并 book\authors\author 和 book\authorlist\author 为同结构序列，规避了因 authors 和 authorlist 的用词不同而误判为不同序列的现象，提高了聚类的准确度。

4.1.2 聚类步骤

考虑到中心点划分聚类方法 PAM(Partitioning Around Medoids) 在文档数 N 和 K 值较大时，计算代价较高，本文选用经典的基于凝聚的层次聚类算法 AGENES (AGglomerative NESting)[14]，按式(1)的 XML 相似性计算方法，对 XML 进行聚类。

输入：(1) N 个待聚类的检索文档；(2) 终止条件 K，K 为划分后族的个数。K 值根据构成分布式环境的机器个数确定，本实验用 3 台机器模拟分布式环境，设 $K=3$。

输出：K 个族的集合，同族内文档相似度高，族间文档相似度低。

具体步骤：

1. 每个 XML 文档自为一 Cluster；

2. 逐个扫描 XML 文档，获取频繁 e 序列集合；

3. Repeat；

4. 由式(1)计算 Cluster 间各 XML 文档间的欧几里德距离；

5. 将距离最近的 Cluster 合并；

6. Until Cluster 数达到目标值 K。

4.2 集合选择策略

文档集合由多个文档组成，如果把一个文档集合看作一个虚拟文档，则可以通过对这个虚拟文档建立索引，实现对文档集合的检索，进而将集合选择问题转化成文档检索问题。但是这样的虚拟文档与真实文档之间存在较大差异，主要有以下几方面。

1. 文档长度不同

文档集合中包含大量文档，因此虚拟文档的长度要远大于单文档长度，而且文档集合的大小差异，导致虚拟文档的长度差异也远大于单文档之间的长度差异，而很多检索方法的效率与文档长度密切相关，文档长度对检索算法的效果有较大影响。而对于多文档的检索，效率问题必须重新考虑。

2. 文档蕴涵主题不同

单文档的内容较为单一，蕴含的主题相对集中，而文档集合包含诸多文档，因此由文档集合构成的虚拟文档所蕴涵的内容更加丰富，一个虚拟文档中可能包含多个主题。单文档中蕴含的内容往往是多个实体，一个主题。而虚拟文档中，主题间关系复杂。

3. 词语分布特征不同

在文档检索问题中，单文档含有的词语相对较少，而在集合选择问题中，由多文档构成的虚拟文档含有大量词语，虚拟文档的词语分布与单文档的词语分布差异较大，例如：某查询关键词不可能在每个单文档中出现，但可能在每一个虚拟文档中出现。

由于单文档检索和多文档构成的虚拟文档差异较大，要想获得较好的检索效果，不能简单地将单文档检索方法用于多文档检索。因此，需要单独研究文档集合的选择问题。

4.2.1 查询与集合相关度算法

在信息检索领域及计算信息重要度的算法中，TF/IDF 理论最为经典。但是传统的 TF/IDF 算法是针对纯文本信息检索的，没有考虑 XML 文档的结构特征。而 XML 文档中，关键字出现在标签或文本中，以及出现在文档的不同深度，区别信息的贡献度是不同的。通常情况下，XML 文档中，越靠近根节点的关键字对区分信息的影响越大，反之越小。出现在结构中的关键字比出现在叶子节点(文本)中的关键字对信息的区分作用越大。因此，在计算 XML 关键字权重时，不仅要考虑关键字频率，还要考虑关键字的属性以及出现的位置。本文在 TF/IDF 基础上，提出了一种综合位置和属性信息的、计算检索关键字和检索集合相关度的算法，以确定文档集合选择过程中的优先顺序，避免检索不相关集合，从而减少检索开销，较大程度提高检索效率。

XML 的数据分 3 类：标签 Element、属性 Attribute 和文本 Text。其中，Attribute 可以看作和所属 Element 有紧密兄弟关系的 Element，因此 XML 数据可简划分为两类，一类是检索目标的具体内容，以 Text 形式存在；另一类是界定检索范围的条件，以 Element 或 Attribute 形式存在。同样，用户输入的关键字按照对检索结果的作用也可分为两类，界定检索范围的结构信息(简记为 S)和检索的具体内容(简记为 C)。

经典的 TF-IDF 词语加权算法为：

$$w_k(d)=\frac{tf_k(d)\times idf_k(d)}{\max_i\{tf_i(d)\times idf_i(d)\mid 1\leqslant i\leqslant n\}} \tag{2}$$

式中，$tf_k(d)$是关键字 k 在文档 d 中的频率。$idf_k(d)=\log(N/n_k+0.01)$，$N$ 为被检索文档总数目，n_k 是包含关键字 k 的文档总数目。式(2)中的分母为归一化处理。显然 $0\leqslant w_k(d)\leqslant 1$。

显然式(2)没有考虑 XML 的结构特征。本文提出一种针对 XML 特征的计算关键字权重的算法，具体说明如下。

按照上述分析，关键字分为两类：结构信息和内容信息。本文从这两个角度出发，分别计算关键字的结构权重 $w_s_k(d)$和内容 $w_c_k(d)$权重。因为 XML 的结构性、关键字出现的深度对结构权重和内容的权重均有影响，所以先给出关键

字的深度权重。

1. 关键字深度权重

在 XML 中，关键字越接近根节点，区别信息能力越强，反之亦然，因此计算关键字的深度权重公式为：

$$dep_k(d)=\alpha\frac{1}{dep}+(1-\alpha) \tag{3}$$

式中，α 取值根据文档结构重要度决定，一般令 $\alpha=1$，dep 是关键字 K 在频繁路径序列中的深度，根节点为 1。

2. 关键字结构权重

XML 文档中，出现在结构中的关键字比出现在叶子节点（文本）中的关键字区分信息的作用大。因此，计算结构信息时，将出现在结构上的关键字单独计算。再根据关键字深度权重，得出综合了深度信息、位置信息以及出现频度的关键字结构权重为：

$$w\_s_k(d)=\frac{\sum dep_k(d)\times idf_k(C)}{\max_i\{\sum dep_i(d)\times idf_i(C)\mid 1\leqslant i\leqslant n\}} \tag{4}$$

式中，$dep_k(d)$ 为式(3)，$idf_k(C)=\log(N/n_k+0.01)$，这里 N 为被检索文档集合 C 中文档的总数目，n_k 是包含了关键字 k，且 k 出现在结构节点上（即属性为 S）的文档数目。

3. 关键字内容权重

对于出现在 XML 文本中的关键字，计算权重时往往忽略了文本与结构的关联关系，直接借用文本信息检索的 TF/IDF 算法。但是，文本出现在不同的元素下，含义不同，这种差异应该反映在权重计算中，式(5)为反映了结构的关键字内容权重的计算公式。

$$w\_c_k(d)=\frac{\sum dep_k(d)\times idf_k(t)}{\max_i\{\sum dep_i(d)\times idf_i(t)\mid 1\leqslant i\leqslant n\}} \tag{5}$$

式中，$dep_k(d)$ 为式(3)，$tf_k(t)$ 是关键字 k 在文档 d 所在 Text 中的频率。$idf_k(t)=\log(N/n_k+0.01)$，$N$ 为文档 d 中包含的总关键字个数。n_k 是包含关键字 k 的 text 节点总数目。式(5)中的分母为归一化处理，显然 $0\leqslant w\_c_k(d)\leqslant 1$。

4. 关键字在文档 d 中的权重

由结构权重 $w\_s_k(d)$ 和内容权重 $w\_s_c(d)$，得到优化后的关键字权重计算公式，见式(6)。

$$w_k(d)=\beta w\_s_k(d)+(1-\beta)w\_c_k(d) \tag{6}$$

式中，β 值根据实际情况调整，本文令 $\beta=0.5$。

5. 关键字在文档集合 C 中的权重

文档集合由文档组成，由式(6)容易得出关键字 k 在文档集合 C 中的权重，见式(7)，其中 n 为文档个数。

$$w_k(C)=\frac{1}{n}\sum_{i=1}^{n}w_k(d_i) \tag{7}$$

6. 查询 Q 与文档集合的相关度

一般情况，查询 Q 由 m 个关键词组成，可得查询 Q 与文档集合 C 的相关度计算方法，见式(8)。

$$\mathrm{Sim}(C,Q)=\frac{1}{m}\sum_{j=1}^{m}w_{k_j}(C) \tag{8}$$

显然，$0\leqslant\mathrm{Sim}(C,Q)\leqslant 1$。

4.2.2 查询策略

查询由 m 个关键字组成，文档集合由 n 个文档构成，由于容易得出关键字在文档中的深度、频度、属性，因此通过上述式(3)一式(6)，容易求出每个关键字在文档中的权重，再通过式(7)，容易得出关键字在集合中的权重，因此通过式(8)，可计算得出查询与集合的相关度。按相关度大小对集合排序，根据查询精度要求，设置相关度阈值，取舍参查集合。

查询具体步骤为：

1. 由式(6)，计算关键字在每个文档 d 的权重

2. 由式(7)，计算关键字在集合 C 上的权重；

3. 由式(8)，计算出查询 Q 与每个集合的相关度；

4. 根据查询 Q 与集合的相关度，排序待检索集合，根据本次查询精度要求，设置相关度阈值 θ，去除本次查询不相关集合；

5. 按照文献[9]给出的关键字构造结构查询语句方法，构造结构查询语句，进行检索；

6. Repeat

Until 所有待查集合检索完毕。

5 实验

5.1 实验设计

实验数据选取 XML 检索领域中常用的测试数据集：Sigmod 数据集，因实验采用 3 台 PC 构成分布式环境模型，故设 $3n$ 个集合数为聚类终止条件。不失一般性，本实验取 $n=1$，即将文档集聚类成 3 个族。因文献[8]中的 SLCA 算法是 XML 关键字检索中效果较好的经典算法，选择 SLCA 算法作为查准率和查全率的对比方法，XMDR 算法中，影响查准率和查全率的关键字构建结构查询语句的算法，笔者在文献[9]中给出了详细的实验结果和分析，故略。本文的主要创新点是采用分布式检索方式，通过集合划分和集合选择策略实现多文档的检索，因此本文实验重点考查单机非分布式和多机分布式的实验比较，重点关注 XMDR 方法的效率。

为验证 XMDR 在多文档检索时的效率，每个文档集合分别抽取出 200 个文件、600 个文件和全部文件（约 1000 个），每个测试用例分别在文档集合包含文档个数为 200，600，1000 时，进行测试。为消除运行时随机误差，每个测试用例测试 5 次，取 5 次平均时间。

用 3 台 PC 机构建分布式环境模型，配置为：CPUP4，2.6GHz，内存 1GB，硬盘 160GB，操作系统 Windows XP。编程语言为 java，编译环境为 Eclipse 3.4.2，数据库为 MySql 5.0。主服务器通过 Socket 方式将检索关键字发送给从服务器，检索结果分两步返回，文档 ID 和结果 LCA 根节点的 DeweyID 通过 Socket 方式返回给终端服务器（检索以该时间节点为结束点）；如用户需要查询结果详情，则详情通过 FTP 方式返回给终端服务器。分布检索架构示意图如图 3 所示。

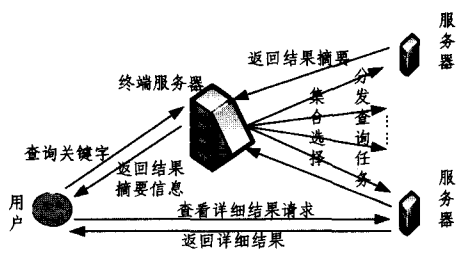

图 3 分布式检索架构示意图

5.1.1 测试用例

根据 Web 调查，用户输入的查询关键字平均个数为 2.4，因此设计测试用例的查询关键字个数为 1～3 个，以反映搜索引擎实际情况。测试用例见表 1，实验结果见表 2。

表 1　测试用例

编号	inputed keywords
QS1	authors articles
QS2	authors Mayford
QS3	articlesTuple James Establishing
QS4	term
QS5	term Information Syste
QS6	term Mathematics General
QS7	author articles
QS8	sectionList Association Rules
QS9	articles index Cathegories

表 2　测试结果

编号 \ 文件数	分布环境检索耗时(ms)			单机环境检索耗时(ms)		
	200 * 3	600 * 3	1000 * 3	600	1800	3000
QS1	6484	18343	30203	16891	51188	84063
QS2	6695	8203	16453	14297	18344	48407
QS3	5571	9610	15899	11406	27500	45140
QS4	3188	8110	16219	4812	12453	29922
QS5	4585	9501	15906	4703	11344	17969
QS6	3383	9470	14258	6156	18031	23922
QS7	6407	14974	30203	17891	43188	87063
QS8	4304	10646	20157	12125	28500	58093
QS9	3973	9003	16609	11922	26172	48109

图 4 给出分布式环境和单机环境下 XML 检索耗时的比较。

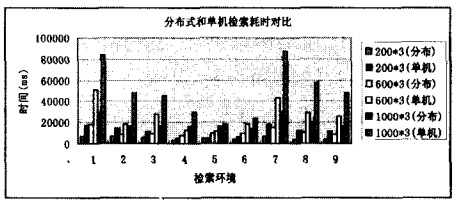

图 4　分布式环境和单机环境下 XML 检索耗时的比较

5.2　试验结果分析

因影响 MXDR 方法检索质量的主要因素是如何由关键字构造准确的结构查询语句，这部分内容笔者在文献[9]中已详细叙述，并给出了有关检索质量(查全率和查准率)的实验数据和详细分析，本文侧重点在于提高检索的效率，因此本文仅针对 MXDR 方法的效率进行实验对比和分析。

聚类后的 3 个文档集：1 号文档集 OrdinaryIssuePage (981 个文档)，2 号文档集 IndexTermsPage(921 个文档)和 3 号文档集 ProceedingsPage(1088 个文档)，共计 2927 个 XML 文档。从每个文档集中分别抽取出 200 个、600 个和全部文件(约 1000 个)，每个测试用例分别在 3 个文档集合包含文档个数为 200，600，1000 时进行测试。每个测试用例测试 5 次，取 5 次平均时间为检索消耗时间。图 4 对比了每个测试项在 6 种不同条件下消耗的时间，从图 4 容易看出，检索文档数越大，分布式检索的优越性越强，尤其是采取合适的聚类方法和集合选择策略时，使得检索集中到部分相关文档集合中，去除了不相关集合，真正减少了检索开支，较大程度上提高了检索的效率。

MXDR 方法提高多文档检索效率的主要原因是：

1. 分布式的检索环境，使检索可与多线程同步进行，缩短了检索时间；

2. 对多文档进行合理聚类，针对检索关键字排除不相关集合，缩小了检索范围，减少了检索开销，较大程度地提高了检索效率；

3. 采用关键字加结构的方法，明确了检索目标，减少了无意义结果的返回，在一定程度上提高了检索效率。

结束语　本文提出一种基于关键字的 XML 检索算法 MXDR，这是一个崭新的检索方法。MXDR 首先用一种兼顾结构和内容的聚类方法对多文档进行分类，并提取类别结构信息，通过分析查询关键字和类别结构信息，确定分布查找策略，再结合查询关键字和 XML 的结构信息，构建结构查询语句，最后通过结构查询系统实现关键字检索。实验表明，基于分布式的 MXDR 关键字构建结构的方法，有较高的查准率和查全率，尤其在检索大量的 XML 文档时，分布式检索方式大大缩短了检索时间，显著减少了检索开销，提高了检索效率。

参 考 文 献

[1] Li Y, Yu C, Jagadish H V. Schema-free XQuery [C]// Proceedings of the Thirtieth international conference on Very Large Data Bases. Volume 30, Toronto, Canada: VLDB Endowment, 2004:72-83

[2] Grust T. Accelerating XPath location steps[C]//Proceedings of the 2002 ACM SIGMOD international conference on Management of data. Madison, Wisconsin: ACM, 2002:109-120

[3] Liu Z, Cher Y. Reasoning and identifying relevant matches for XML keyword search[C]// Proc. VLDB Endow. vol. 1, 2008: 921-932

[4] Theobald M, Bast H, Majumdar, et al. TopX: efficient and versatile top-k query processing for semistructured data[J]. The VLDB Journal, 2008, 17:81-115

[5] Tata S, Lohman G M. SQAK: doing more with keywords[C]// Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data Vancouver. Canada: ACM, 2008: 889-902

[6] Guo L, Shao F, Botev C, et al. XRANK: ranked keyword search over XML documents[C]//Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data. 2003: 16-27

[7] Shao F, Guo L, Botev C, et al. Efficient keyword search over virtual XML views[J]. The VLDB Journal, 2009, 18:543-570

[8] Xu Y, Papakonstantinou Y. Efficient keyword search for smallest LCAs in XML databases[C]// Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data. Baltimore, Maryland: ACM, 2005:527-538

[9] 李霞，李战怀，陈群，等. XML 关键字检索中推断用户需求信息对象的方法 XObject[J]. 西北工业大学学报，2010，4:602-608

[10] Badue C S. Distributed query processing using partitioned inverted files[D]. Federal University of Minas Gerais, Belo Horizonte, Minas Gerais, Brazil, March 2001

[11] Yi J, Sundaresan N. A classifier for semi-structured documents [C]// Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. United States: Boston, Massachusetts, ACM, 2000

[12] Denoyer L, Gallinari P. Bayesian network model for semi-structured document classification[J]. Inf. Process. Manage., 2004, 40:807-827

[13] Cohen S, Mamou J, Kanza Y, et al. XSEarch: a semantic search engine for XML[C]// Proceedings of the 29th International Conference on Very Large Data Bases. Volume 29, Berlin, Germany: VLDB Endowment, 2003:527-538

[14] Han Jia-wei, Kamber M. 数据挖掘概念与技术[M]. 范明，孟小峰，译. 北京：机械工业出版社，2008:261-271