第 34 卷　第 18 期　　计　算　机　工　程　　2008 年 9 月
Vol.34　No.18　　Computer Engineering　　September 2008
·软件技术与数据库·　文章编号：1000—3428(2008)18—0036—03　文献标识码：A　中图分类号：TP311.13

分布式数据库搜索引擎的索引建立和优化

蒋　维，郝文宁，杨晓恕，靳大尉
(解放军理工大学工程兵工程学院，南京 210007)

摘　要：对于使用数据的用户来说，能找到准确的数据且没有遗漏是一件非常困难的事。为了较好地满足用户需求，该文提出了利用分布式数据库搜索引擎架构来实现智能化的搜索和定位。通过建立和优化索引，并使用适当的排序算法，搜索引擎能将最贴近用户需要的结果排在其他结果之前，从而提高搜索引擎的检索效率、查全率和查精率。实验表明，该引擎的查全率为 90.02%，查精率为 89.78%。

关键词：分布式；数据库搜索引擎；索引建立；索引优化

Index Creation and Optimization of Distributed Database Search Engine

JIANG Wei, HAO Wen-ning, YANG Xiao-jia, JIN Da-wei
(Engineering Institute of Engineering Corps, PLA Univ. of Sci. & Tech., Nanjing 210007)

【Abstract】 It is very difficult for the users to find what they need fast and effectively. In order to satisfy the users' requirement, this paper gives a system structure of the distributed database search engine, which implements the intellectualized search and orientation. The construction of the search engine focuses on the establishment and optimization of the index. By the creation and optimization of the index, and using a proper collate algorithm, the search engine can give the users what they need at first. Thus, it can improve the search efficiency, recall and precision ratio. The experiment shows that the recall ratio is 90.02% and the precision ratio is 89.78%.

【Key words】 distributed; database search engine; index creation; index optimization

1　概述

Lucene是一个基于Java的优秀的开源全文本搜索技术框架。按照Lucene的框架规范，扩展Lucene 的功能，可以将Lucene 很好地嵌入到自己的搜索引擎中[1]。

目前国内还没有专门针对分布式数据库的引擎，比较好的像“天网”、“网络指南针”、“木棉”等搜索引擎的索引结构不是基于 Lucene 实现的，它们在实现结果排序时主要采用的 2 种索引相关度算法都不是针对数据库的，没有考虑到专业领域数据的特殊性，因此，本文设计一个基于 Lucene 面向服务的分布式数据库搜索引擎。该搜索引擎提出了一种新的考虑数据位置的索引，同时对二进制流文件也进行了处理，依靠专门领域的术语词典，力求达到很精确的分词，提供对结构化数据(如数据库等资源)和非结构化数据(如文档等数据)的目录式检索和全文检索等功能。这将为各种软件提供底层的数据支持，对资源交流具有重大意义。

2　分布式数据库搜索引擎框架简介

分布式数据库搜索引擎的架构如图 1 所示。本搜索引擎的架构主要分成 6 个层次：系统表示层，业务功能层，安全链接层，业务逻辑层，对外服务层和数据访问层。系统表示层为用户提供统一的检索入口点，即提供一个门户网站给用户进行检索。当用户发出检索请求后，通过 TCP/IP 协议将请求发向检索服务器；在检索服务器上进行处理的业务功能包括索引管理、日志管理、全文检索、目录式检索、结果显示、用户管理、访问控制等 7 个部分。检索服务器通过调用 Web 服务，以自定义的通信协议将检索请求发向各个数据检索点。各个数据检索点根据原始数据库、元数据库和索引文档等将检索结果返回给检索服务器。其中，数据检索点元数据库的建立是为了对各检索节点上的原始数据库进行描述，提供智能的分析和推理；索引文档是针对原始数据库建立的。本文的重点是索引文件的建立和优化。

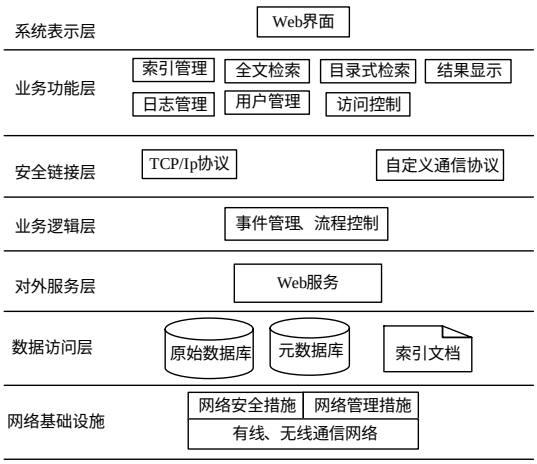

图 1　分布式数据库搜索引擎的架构

基金项目：国家自然科学基金资助项目(70371039)
作者简介：蒋　维(1981－)，女，博士研究生，主研方向：信息系统安全与管理；郝文宁，副教授、硕士；杨晓恕，博士研究生；靳大尉，讲师、硕士
收稿日期：2007-10-25　**E-mail：**lijunling@nju.org.cn

本搜索引擎将 eclipse 作为开发的工具，基于开源代码 Lucene 建立索引。通过对算法的改进，对索引的结果做了进一步优化。从而，实现了智能化的搜索和定位。

3 索引的建立和优化

由于目前索引的建立都是针对文本进行的，但本文提出的分布式数据库搜索引擎是针对数据库进行的，因此，在建立索引前要对数据库进行转换并对数据资源进行分类。

首先，对数据库进行转换。在每个数据节点上浏览数据库，将每张表格读成 xml 文件，如果表中含大字段内容，则通过流方式读成 doc 文件。2 个文件都有相应的命名规则：xml 文件的命名规则是数据库名+表名；doc 文件的命名规则是数据库名+字段名+主键值+表名。

然后，对各检索节点上的数据资源进行分类。通过将数据表读成文本文件后，对文件进行分词和去掉停用词等操作，提取特征项，计算特征项的权重，一个文本就变成了用特征项(词)的权重所表示的一个向量。有了这种表示后，就可以通过前向神经网络的交叉覆盖算法对文本进行学习和分类。覆盖算法主要分为学习和测试两部分。

3.1 基于覆盖算法实现数据的分类

在各个数据库资源分布的节点上，通过对 xml 文件的操作实现各张表的分类。

首先，由某领域的知识专家人工对该领域的数据进行分类，对每个类给出样本。设样本分成 13 大类，样本集合 $X=\{X_1,X_2,\cdots,X_{13}\}$，每个 X_i 表示成 $\{x_i^1,x_i^2,\cdots,x_i^k\}$，通过分词和去停用词等步骤后，将每个文本都表示成一个特征向量。计算样本集合中最大的模 r。

然后将样本中的每个点投影到中心在原点、半径为 2 r 的球面上，取 x_i (i=1,2,⋯,13)中的点 x_i^j (j=1,2,⋯,k)，按照下列公式，计算以 x_i^j 为中心，$d(j)$ 为阈值的覆盖[2]。

$$d_1(j)=\max_{x\in X_i}\{<x_i^j,x>\}$$

其中，<,> 表示内积。

$$d_2(j)=\min_{x\in X_i}\{<x_i^j,x>|<x_i^j,x>\geq d_1(j)\}$$

$d(j)=(d_1(j)+d_2(j))/2$。

通过循环计算 x_i (i=1,2,⋯,13)中所有点的覆盖，从而得到各大类 x_i (i=1,2,⋯,13)的覆盖。

在每个检索节点上，根据每个 xml 文件形成的特征向量，计算它属于哪一大类的覆盖区域，就能判断出其类别。由于特征向量可能属于几个覆盖区域，因此每张表也可能属于多个类别。

3.2 基于 Lucene 的索引文件建立和优化

本文在 Lucene 的基础上进行了二次开发和应用。

3.2.1 索引文件的建立

一般地，Lucene 的索引通过 Field 字段记录，Field 字段包括 Keyword 和 Text 两部分内容。

其中，Field.Keyword记录文件名和文件路径等信息，如表 1 所示；Field.Text则记录分词后索引字段的集合内容，主要包括文章号、出现频率和出现在分词的位置[3]，如表 2 所示。

表 1 改造前 Field.Keyword 记录的字段

名称	内容
FileName	文件名
FilePath	文件路径

表 2 改造前 Field.Text 记录的字段

索引词	文章号	[出现频率]	出现位置
广州	1	[2]	3,6
live	1	[2]	2,5
	2	[1]	2

针对数据库搜索，本系统分别对 Field.Keyword 和 Field.Text 进行了改造。

(1)对 Field.Keyword 的改造

为了让用户通过摘要快速地定位到数据库数据的位置，本系统在 Field.keyword 中增加了数据库名称、数据库访问路径、表名和主键名，如表 3 所示。

表 3 改造后 Field.Keyword 记录的字段

名称	内容
DBInfo	数据库名称+数据库访问路径
TableName	表名
FileName	文件名
FilePath	文件路径
KeyName	主键名

(2)对 Field.Text 的改造

本系统用索引词的“位置特征”代替表中的“文章号”。而位置特征用一个一维向量(a, b)来表示。其中，a 表示索引词出现的文章号，b 表示关键词出现的位置(0 表示出现在 xml 文件中，1 表示出现在 doc 文件中)。

本系统用索引词的“权重”代替表中的“出现频率”。索引词的权重是根据索引词出现的位置(在 xml 文件中，还是在 doc 文件中)以及出现的频率来进行计算的。其中，出现的频率指的是相对词频 tf，即

$tf(t,d)=n/m$

其中，n 表示词语 t 在文档 d 中出现的次数；m 表示文档 d 中去词之后分词的总数。

下面，根据索引词出现的位置以及相对词频来计算索引词的权重。

(1)索引词出现在 xml 文件中

可以将所出现的位置再细分为文件名中和文件内容中 2 种情况进行处理。

1)索引词出现在 xml 文件名中

如果索引词出现在表名中表示此索引词非常重要。因此，取索引词的权重为 $T_1=1$。

2)索引词出现在 xml 文件内容中

索引词在此位置的权重可以用相对词频来代替。即 $T_2=n/m$。

综合上述 2 种情况，索引词出现在 xml 文件中的权重为

$T_{xml}=C_1\times T_1+C_2\times T_2$

其中，C_1和 C_2是 2 个权重系数，要求 $C_1+C_2=1$，且 $C_1\geq C_2$，它们的取值直接影响权重的计算，在后续工作中要适当调整，选择合适的数值。

(2)索引词出现在 doc 文件中

专业领域的文章一般都是结构化的，通常由标题、摘要、关键词、正文组成。在特征提取之前先对文本各个部分的重要程度(在表达文本内容时的贡献大小)做个区分：通常标题部分特征信息的含量都比文章其余部分要大；摘要部分含有的信息量次之；正文的开头和结尾含有的信息量比正文的其他部分要大。因此，采用以下符号来区分索引词出现在doc文件中的位置[4]：

T：表示出现在标题的词语；

S：表示出现在副标题的词语；

A：表示出现在摘要的词语；

F：表示出现在正文开头和结尾的词语；

D：表示出现在正文的其他位置中的词语。

对T, S, A, F和D中的词分别赋予不同的系数I_T, I_S, I_A, I_F和I_D，且$I_T>I_S>I_A>I_F>I_D$，这些系数体现了特征词语在文本的不同区域对主题内容贡献的大小。

综上所述，索引词出现在 doc 文件中的权重为

$$T_{doc}=(I_T\times n_T+I_S\times n_A+I_A\times n_A+I_F\times n_F+I_D\times n_D)/\ m$$

其中，n_T表示检索词在标题中出现的次数；n_S表示检索词在副标题中出现的次数；n_A表示检索词在摘要中出现的次数；n_F表示检索词在正文开头和结尾中出现的次数；n_D表示检索词在正文其他位置中出现的次数；m表示文档去词之后分词的总数。

因此，改造后的 Field.Text 记录的字段如表 4 所示。

表 4　改造后 Field.Text 记录的字段

关键词	位置特征	权重	出现位置
广州	[1,0]	T_{xml}=0.634	3,6
live	[1,0]	T_{xml}=0.581	2,5
	[2,1]	T_{doc}=0.259	2

3.2.2　索引文件的优化

本文在设计索引的存储时考虑检索效率，按照原始数据源中表格的分类就行分别建立索引，属于同一个类别的表建成一个索引文件，控制其大小能在内存中保存，这样用户的查询有针对性。分类建立索引比以前一起建立索引的存储代价要大，因为同一个词可能在多个索引文件中出现，但是结合用户的聚类模型和使用信息对用户提出查询、判断用户可能查询的类型，查询的结果要快得多，本检索系统以检索效率为主要优化目标。

中文全文检索中索引方法有 2 种：正排索引和倒排索引。正排索引是以文档的文章号为关键字，索引表记录了每个文档包含了哪些字，每个字的位置又在哪里。而倒排索引是以字或词为关键字进行索引，相对不断增长的文档数来说字或词的数目是较小的[5]。本文的索引文件采用倒排索引的方式，其中把倒排项中的文档号和出现位置都按递增排列，然后对索引进行压缩处理，可以通过“游程编码”变换把较大的整数变换成较小的整数，再选取一种整数编码方案实现高效的倒排项数据压缩。

4　结束语

针对数据库搜索引擎，本文给出了索引建立和优化的方法。在建立索引文件时，增加了权重和位置特征两部分内容。其中，索引词的位置特征表现为在各个节点上可能出现 3 种不同的位置：同时出现在一条记录的大字段和其他字段中，仅出现在一条不含大字段的记录中，仅出现在一条记录的大字段中。这 3 种情况索引词的重要性不一样。如果索引词出现了上述 3 种情况，首先要求按 3 种情况先后排序，然后针对每种不同情况，根据索引中的权重进行排序。

为索引的建立加入权重等信息，以及将索引词按照分类情况进行分开存放，提高了检索效率。即当用户检索请求收到之后，结合用户聚类模型，判断出与用户请求要求最接近的类，然后到相应类别的索引文件中查找，这样就能大大提高检索效率。但此时付出的储存代价比一般情况要大。如何减少索引存放的代价是后期的研究工作之一。

参考文献

[1] 郎小伟, 王申康. 基于 Lucene 的全文检索系统研究与开发[J]. 计算机工程, 2006, 32(4): 94-96, 99.

[2] 周　瑛, 刘政怡. 覆盖算法在文本分类中的应用[J]. 情报理论与实践, 2006, 29(1): 115-117.

[3] 张晓卫, 朱巧明. 一种基于 Lucene 的 Web 全文信息检索系统的设计与实现[J]. 计算机与现代化, 2006, 22(12): 111-115.

[4] 彭　波. 大规模搜索引擎检索系统框架与实现要点[J]. 计算机工程与科学, 2006, 28(3): 1-4.

[5] 罗　旭. 第四代搜索引擎——主题搜索引擎的设计与实现[D]. 北京: 北京大学, 2001.

(上接第 35 页)

由图 2 可知，目标函数随进化代数的增加而增加，并最后稳定在一个最大值，说明采用遗传算法具有优化基版本任务时限的作用。同时最后一代的最优个体的目标函数值为 0.705 8。这比 FTRMFF 的 25%和 RMEDFFF 的 35%利用率大大提高。说明本文提出的优化容错算法是有效的。

5　结束语

分布式控制系统中的任务有实时性要求，这些任务如果未能在规定的时间内完成，可能会造成严重的后果。为此分布式控制系统应有一定的容错能力以提高可靠性。

本文首先分析了分布式控制系统中任务的特性，给出了任务的模型。在此基础上，结合 EDF 算法、版本复制技术及启发式任务分配策略，给出了一种应用于异构分布式控制系统的容错调度算法。为了提高算法的处理器的利用率，采用基于整数编码的遗传算法对基版本时限进行优化，并给出了选择算法、交叉和变异策略，以提高处理器的利用率。仿真结果表明系统的利用率可达到 70%左右，比 FTRMFF 和 RMEDFFF 的利用率大大提高。这说明本文给出的调度算法是有效的。

参考文献

[1] 秦　啸, 韩宗芬, 庞丽萍, 等. 混合型实时容错调度算法的设计和性能分析[J]. 软件学报, 2000, 11(5): 668-693.

[2] 刘　怀, 费树岷. 基于 EDF 的分布式控制系统容错调度算法[J]. 软件学报, 2003, 14(8): 1371-1378.

[3] Bertossi A A, Mancini L V, Rossini F. Fault-tolerant Rate-monotonic First-fit Scheduling in Hard-Real-Time Systems[J]. IEEE Transactions on Parallel and Distributed Systems, 1999, 10(9): 934-945.

[4] 杨春华, 桂卫华, 计　莉. 基于多处理机的混合实时任务容错调度[J]. 计算机学报, 2003, 26(11): 1479-1486.

[5] Layland J W. Scheduling Algorithms for Multi- programmming in Hard Real-time Environment[J]. Journal of Association for Computing Machinery, 1973, 20(1): 46-61.

[6] 张　彤, 张　华, 王子才. 浮点数编码的遗传算法及其应用[J]. 哈尔滨工业大学学报, 2000, 32(4): 59-61.