基于 Lucene.Net的分布式全文检索系统

谭文堂　贺明科　李　阜
(国防科学技术大学信息系统与管理学院　湖南 长沙 410073)

摘　要　随着互联网的发展,现代信息量急剧增加,人们对于信息的检索要求越来越高,一个好的检索系统必须具有较快的检索速度和较高的查准率。针对海量文本数据提出一种基于 Lucene.Net全文检索引擎构建的分布式全文检索系统,使用.NET Remoting实现分布式的全文索引与全文检索,具有较好的扩展性和很快的索引与检索速度,并成功地将该技术应用于军队某部信息管理系统,取得了很好的效果。

关键词　Lucene.Net;.NET Remoting;分布式;全文检索;全文索引

DISTRIBUTED FULL-TEXT SEARCH SYSTEM BASED ON LUCENE.NET

Tan Wentang　He Mingke　Li Fu
(School of Information Systems and Management, National University of Defense Technology, Changsha 410073, Hunan, China)

Abstract　Along with the development of Internet, the quantity of modern information increase rapidly, the requirement of information searching from people becomes more and more exigent, an excellent searching system must search quickly and accurately. This paper proposes a distributed full-text search system based on Lucene.Net full-text search engine for massive text data. We choose .NetRemoting framework to implement distributed full-text index and search, this system has excellent expansibility and high speed in indexing and searching. Our approach is validated in the information management system at a military dept.

Keywords　Lucene.Net;.NET Remoting;Distributed;Full-text search;Full-text index

0 引　言

随着现代社会信息量的急剧增长,各种以 Internet为载体的电子信息愈来愈多,如何有效、快速、准确地在海量信息中查找所需要的信息,已经成为人们的重要需求。全文检索是一种非常有效的信息检索技术,它使人们可以在各种文本中搜索包含指定关键字的文本。全文检索极大地提高了从海量数据中检索或查找特定信息的速度和效率。

全文检索经历了单机、C/S结构到 B/S结构的发展历程。随着 Internet的飞速发展,基于 B/S结构的全文检索技术正得到日益广泛的应用。相对 C/S结构来说,B/S具有瘦客户机、易于升级与管理、具有较高的鲁棒性能等特点,但是很难实现复杂的业务和满足文本处理、分析等需求。

本文基于 Lucene.NET和.NET Remoting研究并实现了一种面向海量文本数据的分布式全文检索系统,把 Lucene.NET的索引和检索功能封装为.NET Remoting对象,使用.NET Remoting对分布的远程对象进行集成,实现分布式的全文索引与检索。实验说明本系统实现了海量文本的快速索引和检索,适合于门户网站及部门间等的海量文本资源的检索。通过扩展,系统可以适应 C/S和 B/S这两种应用需求。

1 Lucene.Net全文检索引擎

Lucene.Net是基于 Apache基金会 jakarta的项目 Lucene的二次开发,目的是能够在.Net环境下应用 Lucene的强大的全文索引和检索功能。它提供了简单的函数接口,可以方便地嵌入到各种应用中实现全文索引与检索功能。

1.1 Lucene.Net的组成

图 1为 Lucene.Net组成结构图。

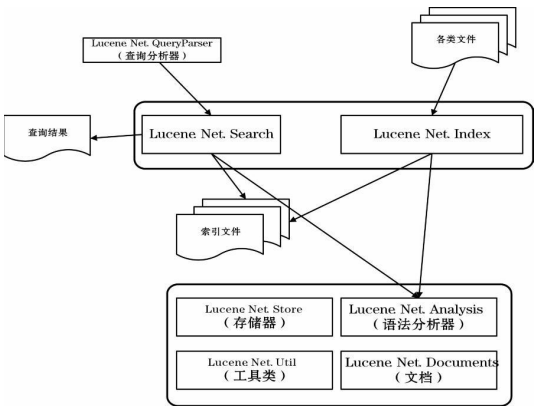

图 1　Lucene.Net组成结构图

Lucene.Net.Index模块　主要实现索引文件的读写接口,通过该模块可以实现索引文件的创建和添加、删除以及读写等。

Lucene.Net.Search模块　主要提供了检索接口。通过该模块可以输入检索条件,得到查询结果集,与 Lucene.Net.QueryParser

收稿日期:2008-05-10。谭文堂,硕士生,主研领域:信息资源管理,分布式系统。

配合还可以支持查询条件间的与、或、非、属于等复合查询。

Lucene. Net. Analysis模块　主要用于分词。分词的工作由 Analyzer的扩展类来实现, 由于 Lucene. Net自带的 StandardAnalyzer类的分词算法比较简单, 只实现了简单的二元分词, 其效果较差, 我们需要实现自己的中文词法分析器。

Lucene. Net. Document模块　封装了对文档对象的定义, 是对所有文档的抽象。

Lucene. Net. QueryParser查询分析器　提供检索接口, 把用户的输入转化为 Query对象以便查询。

1.2　Lucene. Net的优点

Lucene. Net由于其优秀的性能在各个领域都得到了广泛的应用[5]。除了高效快速, Lucene. Net还具有以下突出的优点[4]:

(1) 索引文件格式独立于应用平台, Lucene. Net定义了一套以 8位字节为基础的索引文件格式, 使得不同平台的应用能够共享建立的索引文件。

(2) 增量索引和批量索引, Lucene. Net可以进行增量的索引, 对于大量数据进行批量索引, 并且提供了用于优化批量索引和小批量的增量索引的接口。

(3) Lucene. Net并没有指定数据源的格式, 而是提供了一个通用的结构 (Document对象) 来接受文件的输入, 因此输入的数据源可以是: 数据库、Word文档、PDF文档、HTML文档。

(4) Lucene. Net支持多个字段的索引, 通过控制要索引的域, 可以索引文档的任何属性, 包括日期、作者、摘要等, 并且可以选择是否对某个字段进行词法分析, 即分词[1]。

2　. NET Remoting框架

. NET Remoting是一个丰富的、可扩展的框架, 它使得处于不同 AppDomain、不同进程和不同机器上的对象可以实现无缝通信。它可用于网络上不同计算机的基于 CLR的不同应用程序之间的通信, 也可用于在相同计算机上基于 CLR的不同应用之间的通信。因为可能使用不同的传输协议 (例如 SOAP和 TCP), 应用程序可以使用二进制编码, 也可以使用 XML文件, 因此这是一个非常灵活的架构。其框架如图 2所示。

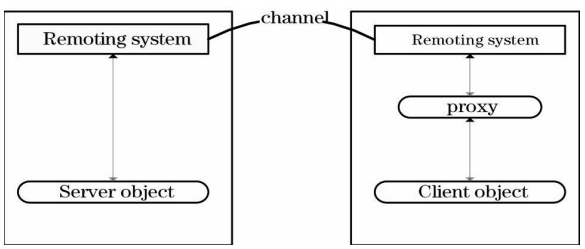

图 2　. NET Remoting框架

. NET Remoting中通过通道 (Channel) 来实现两个应用程序域之间的对象通信。首先, 客户端通过通道访问服务器端对象, 以获得服务端对象的代理。服务器端对象也就是通常所说的远程对象。对于客户程序来说, 代理提供了与远程对象完全一样的方法和属性。当代理的方法调用时, 就会创建消息, 然后将这些消息串行化并发送到客户通道中。通过代理, 客户端应用程序就可以如同使用本地对象一样来操作远程对象。. Net Remoting包括的核心元素有[5 8]:

应用程序域　应用程序域是. NET运行库的逻辑进程表示, 任何实际的操作系统进程可以包含多个应用程序域。

通道　用以实现应用程序域之间的通信, . NET Remoting通过通道穿越. NET Remoting边界传输序列化的消息对象。. NET Remoting基础设施提供了两种类型的通道, 可以用于为分布式应用程序提供传输机制: 支持 TCP协议的 TCPChannel和支持 HTTP协议的 HTTPChannel。

远程对象　是指从 MarshalByRefObject类派生, 且运行于调用方应用程序域之外的对象。在构建多层分布式应用程序时, 该对象用于实现业务逻辑层。

代理对象　作为远程对象的代表, 代理对象确保对代理进行的所有调用都能够转发到正确的远程对象实例, 并将结果返回到客户端。客户端通过调用代理对象实现对远程对象的访问。

3　系统设计与实现

分布式全文检索系统包括分布式索引、分布式检索和负载均衡几个部分, 本文开发的系统需要对各种文档、资料进行全文检索, 所以文档必须同时考虑在文件服务器和数据库里存储, 这样使系统具有高度的扩展性, 充分发挥软硬件的性能, 实现快速索引和检索。系统利用 Lucene. Net优秀的索引和检索性能实现检索和索引的远程对象, 即把检索和索引功能实现为. NET Remoting远程对象, 远程对象分布在各个索引引擎和检索服务器上, 系统通过. NET Remoting对各远程对象进行调度和集成。

系统结构如图 3所示, 系统包括三个服务。

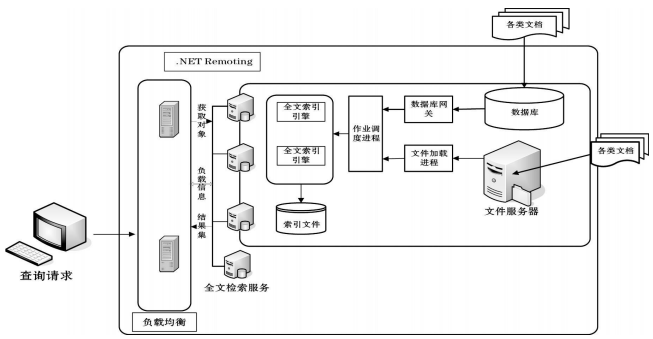

图 3　系统结构

3.1　全文索引服务

全文索引服务对数据库和文件服务器里的文档进行索引, 建立索引文件, 主要包括:

数据库网关　数据库网关定时从数据库里把存储的各类文档检索出来, 提交给全文索引引擎, 其中包括文档的各种属性信息。全文索引引擎对各种文档建立索引。

文件加载进程　文件加载进程定时从文件服务器的目录加载新的文档, 提交给全文索引引擎。

索引作业调度进程　索引作业调度进程对数据库网关和文件加载进程提交的索引作业进行调度, 它保存当前各个索引引擎的负载信息, 把索引作业分配给当前任务最少的索引引擎, 有效利用各个索引节点的硬件性能。

全文索引引擎　如图 4所示, 引擎是分布在各个索引服务器上的远程对象, 负责对提交的各类文档建立索引。引擎采用

Lucene.Net实现，引擎对文档及其各种属性信息建立索引。引擎由索引作业调度进程调度，该进程拥有索引引擎的远程代理，可以动态地调用服务器上的远程索引对象对文档进行索引。

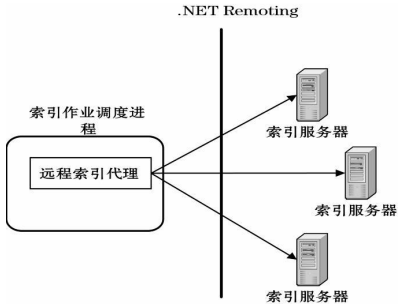

图 4 全文索引引擎示意图

3.2 全文检索服务

全文检索服务接收客户端的检索请求，检索各个索引节点的索引文件，合并检索结果并对结果进行排序。如图 5所示，全文检索服务是分布在各检索服务器上的.NET Remoting远程对象，当用户发送检索请求时，负载均衡进程选择负载最轻的服务器，获取其上的远程检索对象进行检索，该远程对象检索各个索引节点，把检索结果返回给用户。

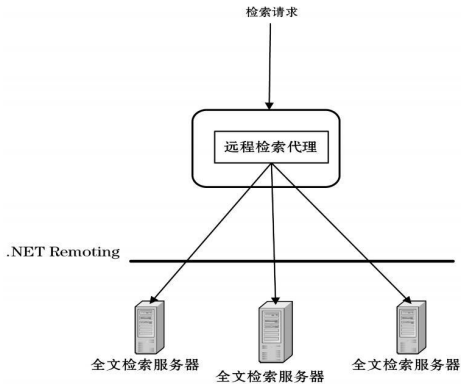

图 5 全文检索服务

这里涉及一个检索的并行性问题，有以下两种选择：

（1）把检索请求分发给各检索服务器让每个服务器分别在一个或者几个索引节点检索，再将检索结果进行合并。

（2）把检索请求分发给负载最轻的检索服务器，让其在每个索引节点检索然后把结果返回给用户。

第一种方案对单个检索请求的检索速度比较快，但第二个方案对于大量并发检索请求的检索速度比第一个快。如果主要要求单个检索的速度，可以选择第一种方案，在所开发的系统中，由于要考虑大量并发的检索请求，在该系统中主要采用第二种方案。

3.3 负载均衡服务

负载均衡服务负责对检索进行调度，它保存当前系统的负载信息，自动把检索请求分配给负载最轻的全文检索服务器。负载均衡服务拥有这些信息：当前可用服务器队列，各服务器平均检索速度，各服务器最近一次执行任务的时间。

本系统基于这样的均衡策略：

（1）接收检索请求，从当前可用服务器队列中找到空闲时间最长的服务器。

（2）从（1）中得到的服务器中选择平均检索速度最快的服务器，若平均检索速度一样，则随机选择其中一台服务器。

（3）从（2）中选择的服务器上获取远程检索对象，发送检索请求。

（4）修改所选服务器的最近一次执行任务的时间。

3.4 系统运行流程

系统由两个相对独立的流程组成：

索引流程 数据库网关和文件加载进程从数据库或者文件服务器加载没有建立索引的文件，提交给作业调度进程，作业调度进程选择一台或者多台索引服务器，获取该服务器上的远程对象，把文件提交给远程对象，由远程对象对其进行索引。其中作业调度进程一直在后台运行，一旦有索引任务提交立即对任务进行分发。

检索流程 首先用户通过客户端提交检索请求，负载均衡服务获取请求，根据各检索服务器的负载情况和检索速度选择合适的服务器，获取该服务器上的远程对象，把检索请求提交给远程对象，由远程对象根据请求检索索引文件返回检索结果。

4 实验结果

系统采用开放源代码的中科院 ICTCLAS分词组件作为 Lucene.Net的词法分析器，并对 ICTCLAS进行了部分优化。部署环境为四台 IBM X3500服务器，其中两台作为检索服务器，另外两台作为索引服务器，为实验系统的性能以三台 PC机模拟检索客户端，PC机上分别运行数量不定的检索线程，通过统计检索的平均时间来衡量系统对并发访问的响应速度。

测试数据是笔者搜集的大量纯文本文件，针对不同文本量进行索引，结果如图 6所示。

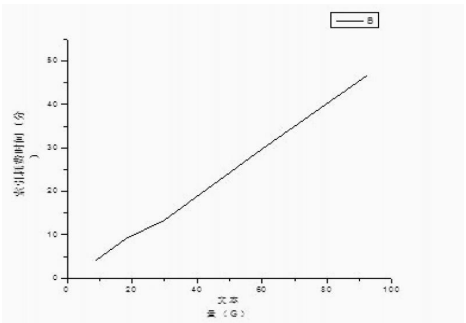

图 6 系统索引速度实验结果

由实验得到系统平均索引速度约为 2.078G/min，并且索引所耗时间几乎呈线性增长，文本量的增加没有对索引速度造成影响。

为测试检索速度，系统使用约 50G文本的索引文件对不同数量的并发客户端进行测试，结果如图 7所示。

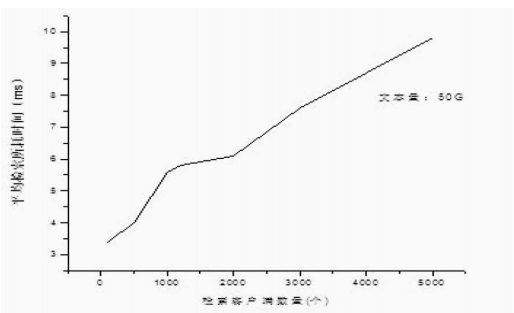

图 7 检索速度实验结果

限于 PC机的性能，PC机上所运行的检索线程不能太多，但

系统对 5000个客户端的平均反应时间小于 10毫秒, 这样的速度已经满足快速检索的需求。

实验结果说明系统具有良好的索引和检索性能, 满足快速索引和检索的需求。系统把分布在不同地方和网络的软硬件资源集成起来, 实现了计算的并行与分布性, 提高了索引和检索的速度。

5 结　论

本文基于 Lucene.Net和. NET Remoting设计并开发了一个分布式的全文检索系统, 创新性地把. NET Remoting应用于全文检索, 实现了一个高效快速的全文检索系统。系统具有高效快速、分布式的特点; 系统充分利用了软硬件资源, 实现快速的索引和检索。存在的问题在于系统未对检索的算法进行优化, 使系统在查全率和查准率上相对一般基于 Lucene.Net的全文检索系统没有明显提高。

该系统在军队某部信息管理系统中得到了很好的应用, 同时该系统也为海量文本检索提供了一个良好的解决方案。

参 考 文 献

[1] 张校乾, 金玉玲, 侯丽波. 一种基于 Lucene检索引擎的全文数据库的研究与实现[J]. 现代图书情报技术, 2005(2): 40-41.

[2] 张英武, 杜凯, 杨树强, 等. 分式海量文本检索系统研究[J]. 微电子学与计算, 2006(23): 20-22

[3] 蔡建超, 郭一平, 王亮. 基于 Lucene.Net校园网搜索引擎的设计与实现[J]. 计算机技术与发展, 2006(11): 73-74.

[4] 赵汀, 孟祥武. 基于 Lucene.Net的中文全文数据库设计与实现[J]. 计算机工程与应用, 2003(20): 179-181.

[5] 马俊, 杨开英, 彭玉卓, 等. NET Remoting与分布式计算[J]. 计算机技术与发展, 2007(2): 118-120

[6] 马保国, 王文丰, 侯存军, 等. 基于. NET Remoting的分布式系统实现[J]. 计算机技术与发展, 2006(3): 50-52.

[7] 林碧英, 赵锐, 陈良臣. 基于 Lucene的全文检索引擎研究与应用[J]. 计算机技术与发展, 2007(5): 184-187.

[8] 王正桓, 蔡明. MS NET Remoting的分布式技术应用研究[J]. 计算机应用与软件, 2005(3): 141-142

(上接第 138页)

```
        if X.count ≥ (|D| + |d|) × s{
        FL_dk = FL_dk ∪ X;
          FL_Dd = FL_Dd ∪ X; } }
    for all X ∈ L_Dk{
        //扫描 D, 判断剩余 L_Dk的项集是否为 D∪ d的强频繁项集
      for all Flag[TID(T)] ≥ k and T ∈ d{
        if X ⊆ T.X.count = X.count + 1; }
        if X.count ≥ (|D| + |d|) × s{
        FL_Dd = FL_Dd ∪ X;
    FL_Dk = FL_Dk ∪ X; }}
```

3 算法实现和比较

通过优化策略 1, MIFUP算法减少了数据库的事务数; 通过优化策略 2在生成 1-频繁项集时, MIFUP算法只需要扫描一次数据库 D或 d。因此从理论上来说, MIFUP的效率远大于 FUP和 PFUP算法。为了进一步比较这两个算法, 下面给出了实际运算比较, 实际运行的硬件环境: CPU是 intel-P4-2.4G, 内存是 DDR-1.0G; 软件环境: 操作系统为 Windows2000 Server, 编程语言是 Matlab7.1。原数据集 D是以布尔矩阵的形式存放, 有 9个项目的 10000条记录的, 新数据集是 500(约 8.5k)条记录。算法 FUP, PFUP, MIFUP执行结果比较如图 1所示。

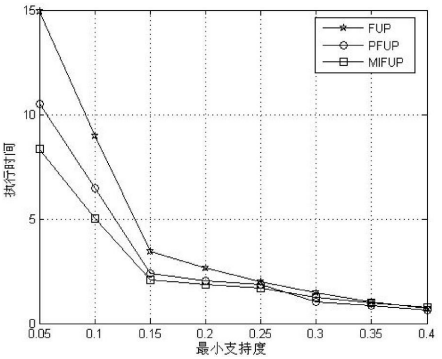

图 1　3种算法执行时间比较

由图 1可知, 总体而言 MIFUP算法的执行时间小于 PFUP算法和 FUP算法。当支持度小于或等于 0.2时, MIFUP和 PFUP算法执行时间相差比较大, 因此当频繁项集数量较大的时候, MIFUP算法的效果更加明显, 且 MIFUP算法更适用于稀疏数据库。当支持度在[0.2, 0.4]变化时, 由于满足该支持度的频繁项集数量不大, 因此 MIFUP算法与 PFUP算法执行速度大致相同。在数据库的关联规则增量更新方面, MIFUP算法具有更小的时间复杂度。

4 结束语

本文在借鉴其他算法的基础上, 提出了一种优化的关联规则更新算法 MIFUP, 与其他算法相比, MIFUP算法减少了扫描数据库 D或 d的次数, 极大地减小了时间复杂度。本文不仅在理论上证明算法的正确性, 而且还通过实验加以验证。但 MIFUP算法只适用于数据库增加的情况, 将来研究重点是拓展 MIFUP算法, 使之能面向增加和删除数据库的情况。

参 考 文 献

[1] Cheung D W, Han J, Ng V T, et al. Maintenance of discovered association rules in large databases: An incremental updating technique[C]//New Orleans, Louisiana: The 12th Int'l Conf on Data Engineering, 1996

[2] Ayan N F, et al. An Efficient Algorithm To Updating Large Itemsets with Early Pruning[C]//San Diego, California, USA: Proc of the 5th Int Conf on Knowledge Discovery and Data Mining (KDD 99), 1999.

[3] 黄德才, 张良燕, 龚卫华, 等. 一种改进的关联规则增量式更新算法[J]. 计算机工程.

[4] 陈劲松, 施小英. 一种关联规则增量更新算法[J]. 计算机工程, 2002, 27(7): 106-107.

[5] 孙士潮, 刘寒冰, 吉立新. 一种高效的关联规则增量式更新算法[J]. 计算机应用与软件, 2007, 24(10): 169-183.

[6] 徐文拴, 辛运帏. 一种改进的关联规则维护算法[J]. 计算机工程与应用, 2006, 18(3): 178-180.

[7] 朱红蕾, 李明. 一种高效维护关联规则的增量算法[J]. 计算机应用研究, 2004, 21(9): 107-109.

[8] Agrawal R, Srikant R. Fast algorithm for mining association rules[C]// In Proceedings of the 20th International Conference on Very Large Data Bases Santiago de Chile, Chile: 1994, 478-499.