基于Z39.50的联机书目检索服务*

杨晓江 张福炎

(南京大学计算机科学与技术系 南京 210093)

摘要 联机书目检索服务是在网络环境下图书馆应当向读者提供的重要服务.基于Telnet或者Web的联机书目检索服务存在检索接口不一致的问题,而Z39.50协议为这种问题提供了解决办法.文章描述了一个基于Z39.50的联机书目检索服务系统.该系统支持中文检索和包括CNMARC在内的多种MARC(machine-readable cataloguing)类型,具有灵活的可配置性和可伸缩性.文章还给出了系统的主要技术设计,并与已有的相关系统进行了简单的比较.

关键词 Z39.50,联机书目检索,MARC(machine-readable cataloguring),PDU(protocol data unit),线程安全.

中图法分类号 TP391

联机书目检索服务是在网络环境下图书馆应当向读者提供的一个重要服务.传统上,图书馆利用Unix主机,通过Telnet提供这种服务.这种方式不但界面不够友好(大多为字符界面),而且由于每家图书馆的检索服务都有自己的检索接口,并要求使用专用的检索程序,给用户带来很大的不便.目前,一些图书馆已经实现了基于Web(或HTTP)的联机书目检索.这种检索由于使用浏览器,用户界面大为改观,但由于HTTP固有的无态性,在Web上只能实现较为简单的检索功能.另外,Web检索虽然只要求用户使用Web浏览器,但用户在检索每一家图书馆时所面临的仍然是各种不同的检索界面.

Z39.50协议[1]是网络信息检索的标准,最初由图书情报界开发,现已成为ISO标准.利用Z39.50提供联机信息检索服务的优点是,Z39.50不但支持各种高级检索功能(如,查询结果的重用和细化),而且能够将每个服务提供者的数据库之间的异构性屏蔽掉,使用户可以使用同一个Z39.50客户检索程序和相同的检索界面去检索分布于Internet上的每一个Z39.50服务器.

Z39.50目前仍然主要应用于图书情报界.国外一些大型图书馆已经或正在积极地支持Z39.50,其中最为普遍的就是提供基于Z39.50的联机书目检索服务.在国内,支持Z39.50也已经被公认为是评价一个图书馆自动化系统的重要指标,但大陆目前还没有一家图书馆、一套图书馆自动化系统支持Z39.50.

我们结合"江苏省高校图书馆自动化系统"的研制,在大陆率先实现了基于Z39.50的联机书目检索系统,包括Z39.50服务器ZServer和客户机ZClient两个子系统,实现了Z39.50版本3的核心功能.ZServer运行于PC/Windows 95/NT和Sun/Unix两种平台之上;ZClient运行于PC/Windows 95/NT之上.经过几个月的试运行,整个系统性能稳定,效率优良,在某些重要的性能指标上,此系统明显优于国外现有的一些典型系统,如,Innopac[2],Sirsi[3],BookWhere[4]等.

---

* 本文研究得到江苏省教委重点攻关项目基金资助.作者杨晓江,1965年生,博士,副教授,主要研究领域为多媒体技术,中文信息处理,网络信息服务.张福炎,1939年生,教授,博士生导师,主要研究领域为多媒体技术,图形处理技术,中文信息处理.

本文通讯联系人:杨晓江,南京210093,南京大学计算机科学与技术系

本文1998-03-13收到原稿,1998-09-04收到修改稿

1 服务器的主要技术设计

1.1 对Z39.50的基本支持

Z39.50是一个庞大而复杂的标准,虽然实现它的所有功能很难,但一个基于Z39.50的联机书目检索服务至少应该满足如下条件:

(1) 支持InitRequest,SearchRequest和PresentRequest这3类请求,即支持Z39.50核心服务.

(2) 支持MARC(machine-readable cataloguing)[5]记录语法,因为MARC是书目数据的标准格式.在我国,至少应该同时支持USMARC和CNMARC两种MARC类型.可用Z39.50支持的UNIMARC替代CNMARC,因为CNMARC与UNIMARC兼容.

(3) 支持RPN(reverse polish notation)查询,因为它是Z39.50标准规定的唯一的必须支持的查询.

ZServer首先实现了对Z39.50的上述基本支持,目前正在对功能作进一步扩充.本文的讨论主要针对上述基本支持.

1.2 PDU的分层处理

PDU(protocol data unit)的处理是实现Z39.50服务中最为复杂的任务之一.借鉴ISO/OSI模型中的分层思想,我们对Z39.50 PDU的处理也采用了分层的模型,图1说明了该模型.

(1) 表示层

表示层封装Z39.50协议规范,提供了表示(构造)各种Z39.50 PDU的功能,它将Z39.50 PDU从ASN.1[6]表示映射到系统的内部表示.

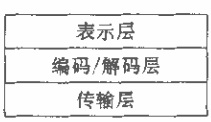

图1　PDU处理的分层模型

为了系统处理上的方便,从PDU的ASN.1表示到内部结构表示有可能增加必要的字段.例如,对于SearchRequest PDU,增加一个字段numberOfDatabasenames用以记录数据库的数量是必要的,因为这从PDU本身在直观上是得不到该信息的,但它却相当有用.

(2) 编码/解码层

编码/解码层通过将数据从本地表示转换成一种公共的标准(编码)或者反过来(解码),协调不同主机或应用程序间的连接.

Z39.50 PDU按照BER(basic encoding rules)[7]进行编码/解码.由表示层得到的PDU在这里被编码,并且通过传输层发送出去;来自客户的PDU在这里被解码,变成PDU的表示层形式.由于编码/解码的使用频率高,内存使用频繁,通过系统调用频繁分配、释放内存显得效率不高.为了提高编码/解码的效率,应该在编码/解码层维护自己的"堆",只有当有必要扩大堆的大小时,才利用系统调用.

(3) 传输层

传输层负责完成PDU在两台机器之间的传输任务.该层通过封装SOCKET运行库中的常用函数,如,*send*(),*recv*(),*accept*(),来实现对PDU的无阻塞(non-blocking)传输.

1.3 书目数据库设计

ZServer后端连接的数据库是一个SQL(structured query language)数据库,ZServer对后端数据库的访问通过ODBC(open database connectivity)来实现.由于提供联机检索用的后端数据库,其信息必须不断更新以反映最新的变化,这样才有最大的服务价值,因此我们的后端数据库直接使用了图书馆自动化系统中的采编流书目数据库.

数据库中与书目检索服务密切相关的表是那些保存MARC数据的表以及与这些表相关的索引表.这里给出一组简化了的保存MARC数据和索引信息的数据库表的设计,它由一个MARC数据表和一个可检词表组成.MARC数据表和可检词表之间的关系是一对多的关系.图2给出了书目数据库的MARC表设计,其中MARC_ID和MARC类型含义是显然的,MARC数据块保存MARC数据.由于MARC数据具有不定长性,我们将一条MARC数据以若干条记录存储;又根据绝大多数CNMARC长度在2 000字节以下这个事实,每一条

记录保存长度为 255×7 的数据块. 图 2 中的 MARC 计数正是为此目的而设计的.

MARC_ID	MARC 类型	MARC 计数	MARC 数据块

图 2　书目数据库的 MARC 表设计

可检词是从书目数据中抽取出来的可供检索的关键词,一般由编目人员在对书目文献进行加工编目时提取而得的,也有的是在有了书目数据库后,成批地从中抽取而得的. 可检词将作为数据库中按关键词检索时匹配成功的依据.

每一个可检词一般都有检索属性与之相关连. Z39.50 标准定义了书目检索属性集 Bib-1,其中 Use 类型包含了大约 100 种检索属性. 由于 MARC 类型的多样性,Z39.50 没有使用 MARC 的字段/子字段代码来注册检索词的属性,而是只使用了一个自然数序列,而且这些属性并不是互相排斥的,这和 MARC 中的字段/子字段类似. 为了避免数据库服务器执行的 SQL 语句过于复杂并影响效率,在为每一个可检词标识字段/子字段名称的同时,还为它标识了一个"本地检索属性",比如,用"01"标识"日期","02"标识"题名",等等. 本地属性一般是对若干个 MARC 字段/子字段的概括. 使用本地属性使得数据库管理系统大大提高了对常见属性进行检索的速度. 图 3 说明了书目数据库的可检词表设计.

MARC_ID	可检词	可检词本地属性	可检词字段/子字段码

图 3　书目数据库的可检词表设计

1.4　查询请求的分析

(1) 属性-字段映射表

为了将接收到的 Z39.50 查询请求中的查询属性映射成 MARC 的一个或多个字段/子字段或者本地检索属性,需要在服务器端建立一张"属性-字段映射表". 映射表应该针对不同的 MARC 类型作不同的映射. 图 4 显示了该映射表的一个部分,其中,左边一栏的数字是 Z39.50 标准中的 Bib-1 属性集的 Use 类型的属性代码,第 2 栏中的"N","U"等表示 MARC 类别(这里分别表示 UNIMARC 和 USMARC),数字"01"、"02"表示 Z39.50 属性被映射成的本地属性代码,而 200f,225f 等则表示映射成的字段/子字段. 在映射表中,Z39.50 中有些查询属性被直接映射到本地属性,而且不管 MARC 类别是哪一种. 如,Bib-1/Use 的属性代码 4(Title)就被直接映射成本地属性代号"02". 但大部分的 Z39.50 属性值需要映射到不同 MARC 的不同字段/子字段,如,Bib-1/Use 的属性值 1(personal name)被映射到 UNIMARC 的字段/子字段 200f,701a,702a,映射到 USMARC 的字段/子字段 100a,110a,111a.

```
1    N:200f,701a,702a;U:100a,110a,111a;
2    03
3    N:200f,710a,711a,712a;U:245a;
4    02
21   04
1003 03
...
```

图4　属性-字段映射表的一部分

映射表以文件的形式存在于服务器端,使系统具有灵活的可配置性和可伸缩性. 当服务系统启动时,映射表被读入内存,一旦接收到来自客户端的查询请求,服务端将依照该映射表和数据库设计,快速地生成 SQL 语句. 如果服务器暂时不支持某些 Bib-1/Use 类型的属性,则映射表中这一行空缺.

(2) SQL 语句的构造

来自客户机的 RPN 查询必须在服务器端被动态地翻译成 SQL 语句. 将 RPN 查询请求转换到 SQL 语句的过程是一个不断消耗内存、组合成更长的新语句的过程,所需内存的大小与数量都不容易预先知道,管理好内存分配与释放是其中一个重要的任务. 因为 RPN 中可能包含着一个递归式的复合查询,转换需要递归地将两个操作数(也是 RPN 查询)所生成的 SQL 条件用逻辑关系组合起来. 需要每次为组合起来的 SQL 条件分配一个新的存储空间,其长度为两个子 SQL 条件的长度之和再加上适当的长度以容纳逻辑连接符、空格等. 这样,分配新的存储空间的任务就陷入一个递归的过程之中,并且只有当整个递归过程完成以后,那些分配的存储空间才能释放.

我们使用链表来管理被分配的内存,即在递归函数中分配存储空间后,将获得的指针加入一个链表,等到 SQL 语句使用完之后,遍历链表并释放所有分配的内存.

除了管理好内存使用,这一阶段的另一个重要任务是解释好来自客户端的任意复杂的RPN查询请求.查询请求中如果包含有服务系统暂时不支持的查询属性,必须将它们过滤掉;如果包含有除了Bib-1/Use类型属性以外的属性,如,截词检索(truncation)和关系检索(relation)等,则应该在构造的SQL语句中反映出来;如果包含有不指明属性的查询,则应该视作关键字查询,即应该构造一个可检索任意属性的查询语句.

1.5 线程安全性考虑

当多个线程对同一个对象进行操作时,必须避免发生冲突而破坏了数据的完整型.例如,数据库前端可能会有多个线程在并发地处理客户的查询请求,系统应保证对每一个客户请求使用不同的数据对象执行操作,否则,某一客户的查询结果则可能被另一客户的查询结果所覆盖.所以,需要根据为每一个连接客户所分配的唯一的标识,来动态地管理一个客户数据区表,每一个客户的数据区主要包括用作执行SQL语句时的绑定参数.

不但如此,很多ODBC驱动程序也并不是线程安全的,即不能利用同一组ODBC环境和句柄来为不同的线程服务.事实上,一个应用程序如果需要由多线程并发地处理SQL操作,可以只分配一个ODBC环境(HENV),但对于每一个线程则必须分配不同的连接句柄(HDBC)和语句句柄(HSTMT).虽然这些分配了的句柄可以被该线程重用,但由于服务器不知道一个客户的下一次请求会在什么时候到来,如果许多客户一直占用着各自的这些句柄,必然导致服务器没有办法为后面连接过来的客户执行查询,这就是说会造成一部分客户占据着一部分句柄不用,而另一部分客户无句柄可用.

我们采用了句柄集中管理的方式来解决上述问题.所有句柄由一个链表来维护,对每个线程实行"需要即申请,用完即回收"的策略.每当某线程需要执行SQL语句时,就由系统首先试图分配一组连接句柄和语句句柄,只有分配成功才能执行SQL操作,并把分配成功的一组句柄加入到句柄表中;一旦该语句执行完毕,立即释放它所占有的句柄,并将它们从句柄表中删除.由于释放句柄速度相当快,所以使用该方法的效果相当令人满意.为了提高线程申请ODBC环境和句柄时成功的可能性,我们设置了一个合理的时间范围(比如10s)和间隔(比如0.5s),在这个时间范围内如果申请失败,则每隔该时间间隔再申请一次,直到超过该时间范围才给客户一个"查询者太多,请稍后再查"的诊断信息.

1.6 查询执行和结果提取

服务器构造好SQL语句后,调用过程*ExecUserSearch*()来执行数据库查询,生成结果集视图,并将记录个数作为SearchResponse的一部分返回给客户.

由于客户提取结果时,指定的记录数目可能很大,系统不可能为每个客户一次性分配足够大的数据绑定区域,所以在处理提取查询结果的请求时,很有可能要分段进行.为了使查询结果能够分段地提取而又不影响速度,我们设计了3个相关的函数,避免每提取一次结果都要重新分配一组ODBC句柄.在申请提取结果之前执行*PreBulkFetch*(),它分配ODBC句柄、从结果集视图中选择记录、执行参数绑定;提取结果使用*ExecUserFetch*(),它根据客户请求的记录位置和个数将记录从结果集中提取出来,并按照客户请求的格式将记录返回给客户端.Z39.50标准版本3还支持分段连续提取查询结果的请求,*ExecUserFetch*()函数被设计成非常适合于处理该请求;提取完结果之后,使用*EndBulkFetch*(),释放ODBC句柄.

由于MARC数据是分段存放的,*ExecUserFetch*()是组合MARC数据最好的地方.如果需要在*ExecUserFetch*()函数中执行复杂的数据库查询以获得更多的记录构造信息,一组ODBC句柄可能会不够.在这种情况下,可以在*PreBulkFetch*()中申请多组ODBC句柄,然后将它们作为参数传给*ExecUserFetch*().

2 结束语

我们用自己设计的Z39.50客户软件ZClient和免费试用软件BookWhere[4]对ZServer进行了测试.测试在一个运行Windows NT 4.0(中文版)操作系统的HP2500服务器(处理器为Pentium 200)上进行,关系数据库使用Microsoft SQL Server 6.5,测试数据库中含有大约120 000条书目MARC数据.经过测试,对于单检索项查询,响应时间一般为1~3s;对于通过布尔运算符连接的含多检索项的查询,响应时间的增加一般与检索项个数的增加成正比.

我们还没有发现支持CNMARC(或UNIMARC)和中文检索的其他Z39.50服务器,ZServer在这一点上首先就优越于Innopac[2]和Sirsi[3]等国外同类系统.例如,将一个带有中文和西文检索项的布尔查询请求提交给Innopac,则检索不到记录,甚至会得到错误信息.另外,ZServer很好地支持带有截词检索和关系检索的查询请求,但是通过对Innopac和Sirsi等Z39.50服务器的查询得知,它们基本上不支持这些检索特性.

参考文献

1 ANSI. Information Retrieval (Z39.50): Application Service Definition and Protocol Specification, ANSI/NISO Z39.50-1995. Bethesda, MD: NISO Press, 1995

2 Innovative Interfaces Inc.. Z39.50 Server (Innopac). Australia, Murdoch University. IP:(134.115.152.130:210)

3 Sirsi Corporation. Z39.50 Server (UNICORN). America, Lehigh University. IP:(128.180.2.47:210)

4 Sea Change Corp. Book where 2000 evaluation copy. http://www.bookwhere.com

5 北京图书馆自动化发展部.中国机读目录通讯格式.北京:书目文献出版社,1991
(Department of Automation Development of Beijing Library. China Machine Readable Cataloguing Format. Beijing: Bibliographic Press, 1991)

6 ISO. Information Processing Systems —— Open Systems Interconnection —— Specifications for Abstract Syntax Notation One (ASN.1). Vienna, VA: Omnicom Inc., 1987

7 ISO. Information Processing Systems —— Open Systems Interconnection —— Specifications of Basic Encoding Rules (BER) for Abstract Syntax Notation One. Vienna, VA: Omnicom Inc., 1987

Online Bibliographic Retrieval Service Based on Z39.50

YANG Xiao-jiang ZHANG Fu-yan

(*Department of Computer Science and Technology Nanjing University Nanjing* 210093)

Abstract The online bibliographic retrieval service (OBRS) is a main service which should be provided by libraries on network environment. Every OBRS provided via Telnet or Web has its own access procedures and a special user interface and query language. The Z39.50 protocol offers a solution to heterogeneity problem. In this paper, a system that provides OBRS based on Z39.50 is described. The system supports Chinese retrieval and multi-MARC (machine-readable cataloguing) including CNMARC, and is highly configurable and scaleable. Main technical design and simple comparison with related systems are given in this paper.

Key words Z39.50, online bibliographic retrieval, MARC (machine-readable cataloguing), PDU (protocol data unit), thread safety.