Intranet上数据库中间件原型的研究与构造*

马 松[1] 盛浩林[2]

[1](中国建设银行上海市分行科技部 上海 200021)

[2](上海亚士帝信息工程有限公司 上海 200233)

E-mail: masong@hotmail.com

摘要 研究和分析了Intranet上常用的数据库接口方法,在此基础上,提出了一种以DB(database)中间件形式在Intranet中应用的新方法.通过对数据库连接和访问机制的管理,改善网络上多用户访问数据库的性能,优化网络传输,并支持与多种数据库的接口.详细地给出了基于该方法而开发的DB中间件的设计思想、实现方法和采用的关键技术.

关键词 Intranet,数据库接口,中间件,Java,JDBC(Java database connectivity).

中图法分类号 TP393

今日企业所拥有的网络大都处在异构环境之中,由以台式系统为中心的客户机和不同类型的服务器构成.系统复杂,软件兼容性和互操作性差,系统升级困难,由此导致企业计算的可靠性和安全性得不到保证,使用不便,系统支持和维护费用居高不下.Internet/Intranet技术的兴起和应用孕育着一场新的企业信息管理的革命.它彻底地改变了企业的原有工作模式和各企业之间及企业与社会之间的联系方式,使企业在其自身发展和社会活动中发挥最大的功效.Intranet由Internet发展而来,它不仅具有传统企业内部网的运作功能和安全性,而且又具有Internet的开放性、灵活性,使企业与社会更紧密地连为一体.它的出现为企业摆脱上述困境提供了一条切实可行的途径.

Intranet主要表现为3个应用层次:信息共享与发布、应用交互与信息存取以及企业业务处理.这三者均与数据库的支持有着密切的联系.随着应用的深入和趋于复杂,对数据库访问时的功能和性能要求就越高,尤其是对上述的第3类应用.当前,Web应用中所采用的方法已不能很好地满足应用的要求.为此,本文提出了一种以DB(database)中间件形式在Intranet中应用的新方法,并开发了一个DB中间件原型DBCM(database connection manager).通过对数据库连接和访问机制的管理,改善网络上多用户访问数据库的性能,优化网络传输,并支持与多种数据库的连接.本文第2、3节将给出DBCM的设计思想、实现方法和采用的关键技术.

目前,实现Intranet上对数据库访问的接口大致可以分为以下3种:[1]

(1) 通用网关接口CGI(common gateway interface)

通用网关接口是一组关于如何在客户端Web浏览器、Web服务器与CGI应用程序之间传递信息的规范,是HTTP(hyper text transfer protocol)服务器与程序进行"交谈"的一种方式.

CGI接口为从HTTP服务器内执行程序提供了一种简单易用的方法,可以使用这种机制来创建动态文档以及作为与HTTP服务器之外的服务程序之间的接口.CGI接口有许多好处,包括服务器软件之间的可移植性,以及大量的公用领域程序(免费程序)和为它设计的开发工具.最大的优势就在于,几乎所有的Web服务器都支持CGI程序.但是,CGI接口也存在着一些不足,其中最主要的就是性能问题,特别是在需要多个数据库连接的多用户应用程序中尤其明显.CGI程序不能被多个客户请求共享,当一个新的请求到达时,即使CGI程序有

---

* 作者**马松**,1973年生,硕士,主要研究领域为数据库,Internet/Intranet.**盛浩林**,1960年生,副教授,主要研究领域为软件工程,工具与环境,微机应用.

本文通讯联系人:马松,上海 200233,桂平路471号10号楼5层上海亚士帝信息工程有限公司

本文1997-09-29收到原稿,1998-01-23收到修改稿

一个实例在运行,也必须再启动一个新的实例.并发请求越多,服务器上创建的并发进程也越多.为每个请求创建一个应用进程会消耗较多时间且需要大量内存,同时限制了应用程序自身可用的资源,最终导致性能降低,增加等待时间.

(2) 专用 API(application programming interface)

针对 CGI 程序的上述不足之处,为了提高 Web 服务器与数据库服务器之间的通信效率和性能,各大 Web 服务器厂商和数据库厂商纷纷推出各自的专用 API.

在 Web 服务器与数据库服务器的连接方案中,Netscape 和 Microsoft 作为 Web 服务器厂商分别推出了适用于各自 Web 服务器的 NSAPI(Netscape server application programming interface)和 ISAPI(Internet server application programming interface).与 CGI 程序是作为独立的进程运行不同,NASPI 和 ISAPI 都是以动态链接库的形式存在的,NSAPI 和 ISAPI 应用程序的效率和性能较之 CGI 程序有大幅度的提高.当然,采用这种类型的 API 也存在一些缺陷,例如,NSAPI 与 ISAPI 相互之间不兼容;它们只能在特定的服务器和操作系统上运行;由于采用了动态链接库的形式,一旦代码质量较差就比较容易造成服务器系统的崩溃;并且进行程序设计时会更复杂.

三大数据库厂商 Oracle,Sybase 和 Informix 提供的 Web 服务器与数据库服务器的接口方案都与各自的数据库产品紧密集成,因而无论是在效率、性能还是安全性方面都达到了较为理想的水平.但也正是由于这种紧密集成,导致了它们只能局限于各自的数据库产品之上,所以兼容性和可移植性差.

(3) Java 数据库互连 JDBC(Java database connectivity)

为了克服 CGI 程序访问数据库效率低和专用 API 可移植性差的缺陷,新近出现的 JDBC 是一种较为理想的接口方案.JDBC 是用来执行 SQL(structured query language)语句的 Java 应用程序接口.它由一组用 Java 程序设计语言写成的类和接口组成,易于向任何关系数据库发送 SQL 语句,支持对多种数据库的访问.运用 JDBC API,只需写出单独一个程序就能够发送 JDBC API 给相应的数据库.Java 语言的中性结构和 JDBC 的开放性使得程序员只需写一次程序就能让它到处运行.JDBC 是一个低层次的 API,它是更高层次 API 的基础.开发人员可以在 JDBC 的基础上,设计出用户更容易理解、便于使用的更高层次的接口和工具.

1　数据库中间件方法的提出

按照通常的方式,在访问 RDBMS(relational database management system)时需要先建立用户连接,然后通过用户连接对数据库进行操作.但是,这种连接方式会受到数据库用户数的限制,即通常的 DBMS 服务器都定义了一个用户数,通过这个用户数限制了同时连接到 DB 服务器的用户数量.当与 DB 连接的用户数量达到这个上限时,新的用户连接请求就会失败.在 Intranet 应用中,网络上会出现大量用户同时访问一个数据库服务器的情况,并且每个用户在一段时间内会保持这种服务连接,从而限制了可用的连接数.但是,当用户在进行客户端的处理时,与 DB 的连接通常是空闲的,如果能够充分利用这些时间,使得这些连接能够为其他用户提供服务,那么,就能大大提高同时服务的用户数.如果在用户不使用这些数据库连接时就断开 DB 连接,虽然可以提高 DB 服务器的多用户服务能力,但是建立 DB 连接通常都是相当费时的操作(可能包括建立某种通信连接、启动服务进程、权限检查等等),这样会降低数据库服务器的服务响应速度,在实际使用时效率较低.我们的设计思想是使用独立的 DB 连接管理进程提供连接管理的服务,客户不是直接与 DB 服务器建立连接,而是通过 DB 连接管理进程建立连接,各种 DB 操作也是通过该 DB 连接管理进程完成的.实际上就是在客户应用与 DB 服务器之间再增加一层中间件,也就是所谓的数据库中间件.[2]

2　数据库中间件原型 DBCM 的设计思想

我们所设计和实现的数据库中间件的基本结构如图 1 所示.其中数据库连接管理器 DBCM(database connection manager)是一个服务进程,是前端的客户与后端的数据库之间进行通信的桥梁.当客户向 DBCM 发出对某个数据库的 SQL 请求时,DBCM 搜索当前可用的与该数据库的连接(DB Connection),通过 DB

Connection 将 SQL 请求转发给对应的数据库服务器. 数据库服务器执行 SQL 语句后,将结果通过 DB Connection 返回给 DBCM,再由它返回给客户. 整个数据库中间件的体系结构采用的是三层(Three-tier)客户机/服务器模型. 中间件与各个客户的数据通信采用流套接字(Stream Socket)机制实现,并且利用了多线程的优点,使得中间件与各个客户的套接字通信能够并发地进行. 中间件与后端数据库的接口使用了 JDBC,并且根据 Internet/intranet 上用户对数据库访问方式的特殊性,采用多客户共享同一个 DB Connection 的机制,提高了数据库访问的效率. 数据库中间件与数据库之间是通过使用 SQL 语言而进行通信的.

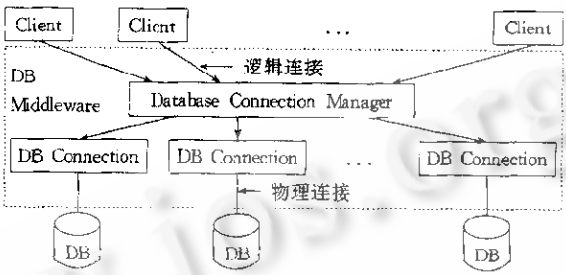

图1 数据库中间件的基本结构

3 数据库中间件原型 DBCM 的实现

根据这个数据库中间件的设计思想,在实现过程中主要运用了下面的关键技术.

3.1 JDBC

JDBC 是一个独立于 DBMS 的接口,即"一个通用 SQL 数据库访问框架",它对于不同的数据库使用了统一的接口,使用 JDBC,程序员只需写一个数据库接口,不需重新编码就能够访问任何数据源. 由于 JDBC 已经逐步得到了各大厂商的认可与支持,也为开发者和用户提供了一条从 Web 服务器访问数据库的捷径,因而可以相信,JDBC 将进一步得以推广而最终成为 Web 服务器访问数据库的事实标准.[3]因此,我们研究和构造的数据库中间件原型就是建立在 JDBC 的基础之上的.

3.2 流套接字

传输控制协议 TCP(transfer control protocol)提供了一种可靠的端对端通信通道,Internet/Intranet 上的客户机/服务器应用程序可以用它来进行通信. 套接字是在网络上运行的两个程序之间的双向通信链路的一个端点. 对应于 TCP 的套接字称为流套接字,它是面向连接的. 服务器应用程序侦听特定端口,等待来自客户的连接请求. 当一个连接请求到达时,客户与服务器之间建立一条专用连接,它们就在这条连接上通信. 在连接过程中,客户被分配一个本地端口号,并将一个套接字绑定到这个端口. 客户通过向套接字写入信息以及从套接字中读出来自服务器的信息的方式与服务器通信. 与此相似,服务器也获得一个新的本地端口号(需要新的端口号是因为,这样,服务器就能够在原来的端口上继续侦听连接请求). 服务器也将一个套接字绑定到它的本地端口,并通过读写这个套接字与客户通信. 客户和服务器必须使用相同的协议,也就是说,它们通过套接字来回传输信息时必须使用相同的语言. 图 2 显示了流套接字的工作流程.

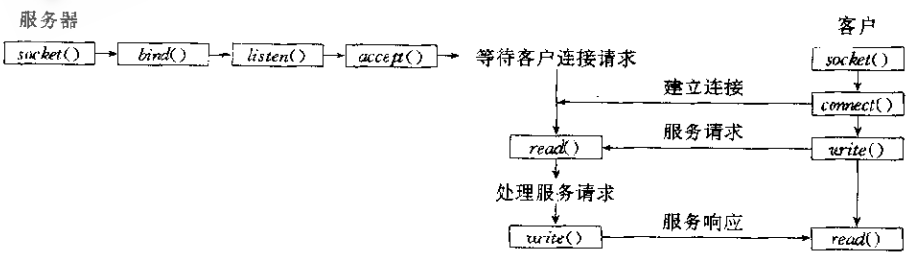

图2 流套接字的工作流程

3.3 多线程

线程是一个程序之中单独的顺序化控制流. 多线程程序设计是指在单个程序中使用多个线程,这些线程在同一时间并发运行,执行不同的任务. 在多线程程序中,各个线程通常需要共享数据. 例如,一个线程向文件写入数据,而同时另一个线程要从这个文件中读数据. 当多个线程共享数据时,需要将线程同步,以达到预期的结果. 如果一个程序中有几个并发线程在竞争资源,就必须运用适当的同步机制来保证公平竞争. 如果每个线程对于有限的资源能够有足够的访问时间来保证合理的进度,这个系统就是公平的.

在数据库中间件中,采用为每个客户提供一个服务线程的方式实现. 在数据库中间件中,主要包括 4 类线程:(1) 监听线程,(2) 服务线程,(3) 空闲管理线程,(4) 状态监控线程. 其中监听线程、空闲管理线程以及状态监控线程是服务器中始终存在的 3 个线程,服务线程是根据客户的请求而产生和消亡的,其数量不定. 图 3 显示了服务线程的工作流程.

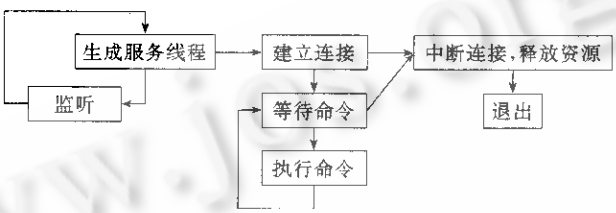

图3　服务线程的工作流程

在编程实现数据库中间件的过程中,特别注意了下面的问题. 由于网络是不可靠的,客户程序同样不可靠,所以加入了充分的出错处理(Exception Handling)机制. 在出错后,可以恢复到某个状态,并释放占用的资源. 失败的客户不会影响服务线程对其他客户的响应. 在多线程程序设计中,服务程序在不工作时应当进入睡眠状态,以释放不必要占用的资源. 另外,尤其要注意临界区冲突的情况. 在 Java 中提供了监视器(Monitors)和 *notify*()、*wait*()方法以及 synchronize 关键字等机制来实施线程的同步.

3.4 多客户共享同一个数据库连接

这种连接方式可以使多个客户共享一个 DB 连接,极大地缓解了客户数量受到 DB 用户数限制造成的问题. 数据库中间件以缓冲池方式管理 DB 连接,减轻了建立和断开 DB 连接时的开销. 将与 RDBMS 接口有关的部分限制在 DB 连接管理中,使得支持异种 DB 更加方便.

这种 DB 连接方案采用在中间件环境中命名 DB 连接的方式定义可用的 DB 连接(逻辑连接),每个连接指定了(RDBMS,DB,USER,PASSWD). 图 1 中客户与 DBCM 之间的连接就是逻辑连接. 在使用这些连接时,只需使用连接的名字而不再提供其他内部的细节. 这种方式有助于减少用户程序对具体环境的依赖性,在 RDBMS 调整时,通过修改中间件环境的定义可以避免不必要的程序修改.

要使多个用户可以共享对一个数据库的物理连接,就要求这些用户与数据库连接时采用的参数(RDBMS,DB,USER,PASSWD)一致,只有连接参数完全一致的两个数据库连接(逻辑连接)才能够共享一个物理的数据库连接. 图 2 中 DBCM 与后端数据库服务器之间的连接就是物理连接.

数据库中间件对外提供的连接名,都是在其自身的环境中定义的,而不是相应 DB 的真实名字. 这些名字称为数据库中间件的逻辑数据库名. 数据库中间件需要管理这些逻辑数据库. 在数据库中间件中,各个逻辑数据库的名字是唯一的,并且对应于某个物理数据库. 每个逻辑数据库包含一组定义信息,包括:物理数据库名、DB 连接用户名、DB 连接用户口令等,可以实现逻辑数据库的访问控制功能,即定义逻辑数据库的访问用户、口令,在客户机创建连接时检查访问控制,避免非法访问.

数据库中间件的客户创建的 DB 连接都是逻辑连接,这些连接在实际操作数据库时映射到数据库中间件与 DB 的物理连接上去,并通过这些连接实际地操作数据库. 这种多个客户共享物理数据库连接的原则是:(1) 能够共享物理连接的用户的 DB 连接在逻辑上是完全相同的,也就是说,以同样的 DB 用户连接到同样的 DB 中去,只有完全相同的连接才能相互替代而不产生逻辑上的问题. 数据库中间件对外提供逻辑数据库服务,只要是连接到同样的逻辑数据库中的用户,其 DB 连接方式就是完全相同的,也就可以共享物理连接. (2) 基于事务处

埋.事务是DB对外服务的基本单位.一个客户在单个事务内必须完全独占一个物理连接,此时不能与其他用户共享物理连接.在事务结束时,如果没有活动的SQL语句,则该客户在该物理连接上已经没有特殊的状态资源了.处于这种状态下的客户相互之间可以共享物理连接.

基于事务处理的特点,对DB物理连接的管理就在事务的开始、终止阶段进行.在事务开始时,需要分配物理数据库连接;在事务终止时,则释放物理数据库连接.

整个数据库中间件的物理数据库连接通过一个缓冲池管理,当需要分配连接时从缓冲池中分配一个连接,当释放连接时则将连接释放到缓冲池中去.其中基本的数据结构如下:

```
// 物理数据库连接信息
class ConnectionInfo {
    Info            dbInfo=null;
    Connection      con=null;
    boolean         isIdle = true;
    java.util.Date idleStartTime=null;
    public ConnectionInfo()   // 构造函数
}
// 管理物理数据库连接的缓冲池
class ConnectionTable {
    static final int MAX_SERVICE=100;
    short serviceCounter=0;
    Service[] serviceArray;
    Vector connectionInformation;
    public ConnectionTable()   // 构造函数,初始化服务数组等
    public Info findDBInfo(Info serviceInfo)   // 根据逻辑服务名检索对应的物理连接
    synchronized public ConnectionInfo findConnectionInfo(Info dbInfo)   // 为逻辑服务分配物理数据库连接
    synchronized public void deleteIdleConnection()   // 释放已超时的物理数据库连接
}
```

客户通过数据库中间件提供的逻辑数据库名建立连接,当客户在这个逻辑连接上发送SQL请求时,DBCM为这个逻辑连接分配对应的可用的物理连接(若当前不存在符合条件的物理连接就新建一个),通过分配的物理连接与数据库服务器通信.空闲管理线程定期在后台运行,检查物理连接的使用状态,若某个物理连接已长时间未用(超时),则断开连接以释放资源.

在具体编程实现这个数据库中间件时,我们选择了Java语言.Java语言是近两年来兴起的一种适用于网络编程的面向对象的程序设计语言.现在,人们已不仅仅将Java语言单纯看作是一种程序设计语言,Java已在引发一场软件革命,正在逐步形成一种新的计算模式——Java计算环境.Java语言的设计目标及特点是:简单性、面向对象、分布性、健壮性、安全性、体系结构中立、可移植性、解释执行、高性能、多线程、动态性.虽然Java语言的风格与C++在很大程度上都具有相似性,但是从复杂性来看,Java语言去除了C++中那些较少使用、难以理解、容易出错的语法元素,而对于内存管理、数据类型的支持、类型转换等各方面却有所增强,所以更加精炼,也比较容易掌握.尤其是Java中不再使用指针,使得系统的安全性加强了.[4]正是因为Java语言具有特别适合于网络编程的特性以及它较之C++语言的优越性,所以选用了Java语言来实现这个数据库中间件.

综上所述,在设计与实现这个数据库中间件的过程中,运用了以下关键技术:①三层客户机/服务器模型,②Java,③SQL,④JDBC,⑤流套接字,⑥多线程,⑦多客户共享同一个数据库连接.

4 小 结

我们所设计和开发的数据库中间件原型针对用户在Intranet上进行数据库访问的特点,采用了多个客户共

享同一个数据库连接的机制,减少了与数据库进行频繁连接和断开的开销,在充分利用资源的基础上提高了效率.由于系统采用Java语言实现,并且利用了JDBC来进行数据库访问,使得系统的健壮性和可移植性得到了保障,可以在多个平台上访问各种关系数据库系统,增强了系统的适应能力.

为了测试该数据库中间件原型的实际运行效果,我们又用Java语言编写了一个客户端应用程序.该程序接收用户输入的各种SQL命令和一些自定义的特定格式的数据库连接与断开指令,将这些请求提交给DBCM执行,然后接收DBCM的运行结果并显示出来.我们采用在联网的机器上同时运行该客户端程序以及在同一台机器上运行该程序的多个实例的方式来增加DBCM的客户数量.客户端程序的操作系统平台采用了Windows 95,Windows NT Workstation,OS/2等,后端的DBMS采用了Windows NT Server上的Microsoft SQL Server,Sybase SQL Server,SCO UNIX平台上的Informix,OS/2 Warp Server平台上的DB2等.通过测试客户端应用程序对SQL命令的响应速度,我们发现当客户数量较少时,采用DBCM情况下的响应速度与不采用DBCM情况下的响应速度相差无几,有时甚至更慢一些.这是因为DBCM管理逻辑连接与物理连接需要花费时间,而各客户逻辑连接之间几乎没有可以共享的物理连接.但随着客户数量的增多,采用DBCM所带来的性能上的优势逐渐显示出来.在我们的测试中,当客户数达到20个时,采用DBCM对SQL命令的平均响应时间比不采用DBCM时缩短约7%,当客户数达到50个时,平均响应时间缩短了约16%.由此可见,客户数量越多,采用DBCM所带来的性能提高就越明显.

概括起来,该数据库中间件的原型完成了下列功能:(1) 有效改善Intranet上多用户对数据库的访问性能;(2) 减少网上信息的传输;(3) 优化了网络传输性能;(4) 支持与多种关系数据库的连接;(5) 既支持类似Web中的无状态和一次连接,也支持应用有状态和持续性的连接;(6) 为Intranet应用中与数据库的连接提供一条功能强大而又灵活高效的途径.

参考文献

1 马松.Internet上的数据库接口.见:Intranet国际研讨会论文集.上海,1997.24～36
(Ma Song. Database interface on Internet. In: Proceedings of the International Seminar on Intranet. Shanghai, 1997. 24～36)

2 马松.Internet/Intranet上数据库接口及数据库中间件原型的研究与构造[硕士学位论文].华东理工大学,1997
(Ma Song. Internet/Intranet DB interface & the research and construction of DB middleware [M.S. Dissertation]. East China University of Science and Technology, 1997)

3 Patel P, Moss K. Java Database Programming with JDBC. Scottsdale: Coriolis Group, 1996

4 Cornell G, Horstmann C S. Core Java. Palo Alto, California: SunSoft Press, 1996

Research and Construction of a Database Middleware Prototype on Intranet

MA Song[1]　SHENG Hao-lin[2]

[1](*China Construction Bank, Shanghai Branch　Shanghai*　200021)

[2](*Application Solutions & Technologies Inc., Shanghai Corporation　Shanghai*　200233)

Abstract　In this paper, the common database interface methods on Intranet are studied and analyzed. Based on that, a new method using DB middleware under the Intranet environment is posed. The key point is to provide an efficient and powerful mechanism to manage database connection and data access to meet the needs of application requirements. It will improve the network performance for multi-user accessing database and optimize network transmission. It can provide the interface with the various RDBMSs (relational database management systems). The design considerations, the implementing and the critical techniques used are given and explained in detail in this paper.

Key words　Intranet, database interface, middleware, Java, JDBC (Java database connectivity).