构建基于 Windows DNA 的三层应用

李学军 申瑞民 (上海交通大学计算机系远程教育研究室 200030)

摘要:本文回顾了应用模型的发展过程,介绍了当前三层应用模型的结构及其优势,以及如何在 Windows 平台上构建基于这一新的模型的应用,并给出了一个实例。

关键词:三层应用 3 - tier Client/Server COM/DCOM

1. 应用模型的发展

最初的计算机应用是一种基于主机/终端模式的计算模型,应用中几乎所有的计算都由主机来完成,终端只是作为一种输出设备。80 年代末,针对这种体系结构的问题与不足,人们提出了客户/服务器(Client/Server)结构,这种模型逐渐得到了广泛的应用。随着应用系统的大型化及基于 Internet 的应用发展的要求不断提高,这种两层结构(2 - tier)的缺陷和不足越来越明显,人们又提出一种三层(3 - tier)的应用模型。

任何一个应用,从 PC 机上的报表程序到大型机上的计算,都由三个部分组成:用户界面部分(表示层),应用逻辑部分(应用逻辑层)和数据访问部分(数据访问层)。表示层的功能是与用户交互,应用逻辑层进行具体的运算和决定程序的流程等,数据访问层维护和更新应用程序的数据。

(1)两层应用模型。在 Client/Server 型的两层式应用中(如图 1 所示),表示层和应用逻辑层被组合在一起,运行在客户端,通过网络连接访问远端的数据。借助于 API 接口例如业界标准的 SQL 语言,客户端的应用组件从数据库中读取数据,执行程序的运算逻辑,然后把数据送回数据库。这种应用模型比较适合于小规模的、用户较少(<100)、单一数据库且有安全、快速的网络环境下运行。由于当前的开发工具大都支持快速的 Client/Server 应用的开发,因而它成为网络应用的主要方式。在有众多用户、多数据库、且非安全的网络环境下(例如 Internet),两层的应用就有明显的局限性(如图 1):

服务器端的数据库必须同每一个活动的客户保持连接,这些连接消耗了大量的运算资源,并且随着客户数目的增加,性能不断下降。

多用户、多数据库的连接存在死锁和系统崩溃的潜在可能。在 Internet 这样的大规模分布式环境下锁住一项数据以防止一些用户访问造成其他用户长时间的等待。

两层模式下的安全管理不适合于非局域网环境。由于事务处理逻辑驻留在客户端,所以这种基于用户授权方式的安全管理下,一旦用户拥有了某项权限就可以为所欲为,即可以绕过客户端的应用逻辑直接操作数据。

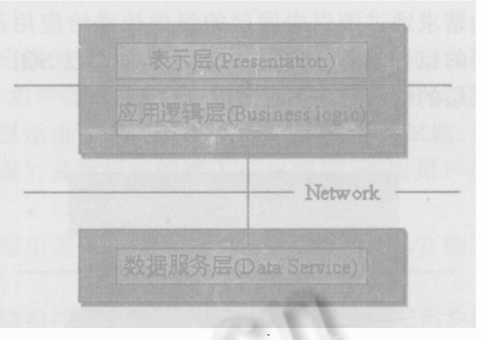

图 1 两层(Client/Server)应用模型

两层应用的代码难以得到重用,因为这种应用(用户界面和程序逻辑在一起)和具体的数据紧密联系在一起。重用只能以 Copy/Paste 源代码的形式而不是二进制组件的形式来实现。

由于应用逻辑全部驻留在客户端,因而当应用环境发生变化需要改变事务逻辑时,每个客户机上的程序都需要更新,给系统的维护和管理造成了一定的困难。

随着应用系统越来越复杂,客户端应用程序变得越来越庞大,对客户机的处理能力要求越来越高,成为所谓的"胖客户机"。

为了缓解以上的各种压力,一些服务器端的数据库系统支持编写"存储过程",使得拥护能够一次执行存储在远端数据库服务器中的一组 SQL 语句。客户端的应用可以调用这些存储过程,向它传递参数,并把执行的结果取回。存储过程大大降低了网络通信的开销,提高了

© 中国科学院软件研究所 http://www.c-s-a.org.cn

安全性和系统代码的可重用性。和把表示层与应用逻辑层组合在一起的两层模型相比，这种两层半方式把一部分程序逻辑移到了服务器端，从而提高了系统的可扩展性、安全性和可重用性，但这几种性能的提高是很有限的，例如不同数据库厂商的存储过程是不兼容的，而且缺乏二进制组件即插即用的优点，因而它的重用就受到了很大的限制。

(2)三层应用模型。就在一两年以前，从事多层应用的开发人员还很少，只有不多的工具支持这种模型上的开发。现在它正在逐渐成为主流，这其中两个因素起了关键性的作用：可扩展性的要求和 Internet 的发展。两层应用可扩展性差，且对客户端的要求越来越高。而 Internet 的发展要求应用在"瘦客户机"上运行，因而程序逻辑就必须与用户界面分开，从而使三层应用成为必要。在三层结构下，表示层(Presentation)、应用逻辑层(Business logic)、数据服务层(Data service)被分割成三个相对独立的单元(如图 2 所示)。表示层负责与用户交互并把相应的请求通过调用中间层的组件传递给应用逻辑层。应用层的组件执行具体的事务逻辑并通过 SQL 等方式向第三层的组件提出数据或其他资源请求。

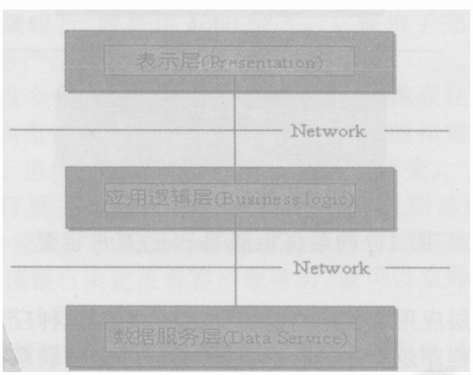

图 2 三层应用模型

三层体系结构常被称做 Server - Centric，因为应用逻辑是运行在中间层的服务器上，与用户界面和数据的访问相对独立。尽管没有要求这三部分必须运行在不同的机器上，一般情况下，表示层在客户端的应用如浏览器中运行，数据访问也在专用的数据库服务器上运行。这种分层方式带来了诸多的优点：

应用逻辑集中放置在服务器上由所有的用户共享，使得系统的维护和更新变得简单，当事务逻辑发生变化时，只需更新服务器上相应的应用逻辑组件，之后所有的客户就可以使用新的事务处理逻辑。避免了客户端应用程序版本控制和更新的困难。

在应用逻辑层，开发人员可以利用 VB、VC 等常用的开发工具开发可重用的二进制组件，而不是编写存储过程。而且这些组件可以镜相到多台机器上同时运行，从而分担多用户的负载。应用程序组件可以共享与数据库的连接，数据库服务器不再是为每个活动的用户保持一个连接，从而降低了数据库服务器的负担，提高了性能。

安全管理可以基于组件来授权而不是授权给用户，客户不再直接访问数据库，提高了安全性。

2.微软平台下三层应用的开发

在 Windows DNA 平台上，软件的开发是基于 COM/DCOM 的组件的开发。应用逻辑层的代码以可重用的二进制组件的形式存在。为有效地管理和利用这些组件，微软推出 Microsoft Transaction Server (以下简称 MTS)。如图 3 所示，MTS 为构架和分发基于 COM/DCOM 技术的三层应用提供一个 Server 端的运行环境。在这种应用中应用层的组件在 Server 上 MTS 的控制下运行。表示层的组件，包括编译过的应用或 Web 页面及脚本程序通过 DCOM 调用应用层的程序(以组件的形式存在)，完成相应的功能。应用层的组件通过 MTS 访问各种不同的数据源，包括数据库，文件系统，甚至 Mail 系统，MTS 提供了包括数据连接缓冲(Connection Pooling)、线程缓冲、锁管理、事务管理机制等多种自动的服务。

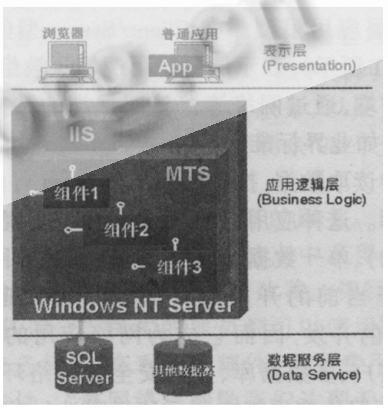

图 3 微软平台下的三层应用结构

开发一个基于 Windows DNA 的三层应用一般要经过以下几个步骤：

(1)理解应用环境需求，将其划分成表示层、应用逻辑层和数据服务层。层次划分是设计过程中很重要的一

© 中国科学院软件研究所 http://www.c-s-a.org.cn

个步骤，将直接影响它的性能和扩展性等多个方面。

表示层主要完成与用户的交互，对用户输入的分析及响应，主要通过在浏览器中运行的活动页面（包括 HTML、DHTML、Scripting、Java applet 及 ActiveX Control）或由 VB 等其他高级语言所编写的界面程序来实现。

应用逻辑层实现具体的事务逻辑，通过将它划分成多个相对独立的组件（模块），封装内部的实现细节，提高它的安全性和可重用性。这些组件可以用支持 COM/DCOM 的多种开发工具来实现。

数据服务层提供数据的存储和访问，可以针对不同的应用选取不同的数据源。

（2）实现应用逻辑组件。为了将来在 MTS 中分发和运行，利用 MTS 的事务管 理等各种服务，在编写这些组件时需要遵循一些特定的规则，例如及时释放资源等。

（3）打包和安装组件。这些应用逻辑组件不仅要在 Server 端安装和配置，还要在客户端注册，以便客户端的应用能够通过 DCOM 访问它们。MTS 为此提供了打包和发行 Server 端组件的工具。

应用层组件的开发可以采用诸如 Visual Basic、Visual C++、Delphi、Visual J++ 等多种成熟的开发环境。与通常的两层应用的开发相比，开发人员 只要遵守几个简单的规则，就可以开发出能够在 MTS 环境下运行，从而享有它的各种服务的组件来。当表示层要调用这些组件时，客户端的 DCOM 会把这些请求自动路由到应用层中的 MTS 上来，由它提供统一的服务。由 MTS 所提供的这些服务使得开发人员不需要掌握服务器端编程的许多复杂接口，只要编写面向单用户的组件，就可以直接在 MTS 中发行，自动支持多用户的访问，从而使得许多桌面应用很容易地转化成服务器端的方案。

3. 一个实例

本例是一个用于在 Internet 上进行考试的智能试题系统，用户可以在任何地方通过浏览器阅读试题进行考试，能够根据用户的不同智能改变试题的内容和表现形式（生成同等难度的不同试题，改变选择题选择支的顺序等）。试题存放于数据库服务器中，考试时送到客户端浏览器中，考试的结果送回服务器端，客户端还可以查看成绩，统计结果等。运行的平台客户端是 IE，服务器端包括 IIS、MTS 和 SQL Server。采用浏览 器作为客户端的界面使得使用本系统的用户的位置不受限制。为保证数据的安全性，试题等数据不是以页面的形式返回，而是由页面中运行的 ActiveX 组件来读取和返回，从而避免了在浏览器 Cache 中遗留等一些安全隐患。试题脚本的生成，成绩的记录统计等由 Server 端专门的组件来实现。于是整个应用被划分成上述的三层结构（如图 4 所示）：

图 4 智能试题的三层结构

客户端是在浏览器中的页面，由 HTML、Scripting、ActiveX 组件等部分组成，负责读取用户身份认证信息，解释和显示由应用逻辑组件所送回的数据（试题、成绩或统计数据），并对用户的输入作出响应，送回用户的答案等。

在应用逻辑层，是在 MTS 中运行的由 VB 编写的组件，包括：

试题组件，负责生成试题脚本、解释匹配用户的答案等；

用户组件，负责进行用户的身份认证等；

成绩组件，负责生成用户成绩，进行统计和分析等。

数据服务层采用 SQL Server，存放用户及试题信息等。

随着即将推出的 NT5.0，微软正在极力地推广他的基于 DCOM 的分布式解决方案，由于 Windows 平台已被大多数的用户所接受和使用，这种 Windows DNA 下的三层模式必然会得到广泛的应用。

参考文献

[1] How Microsoft transaction server changes the COM programming model, Microsoft system journal 1998.1
[2] Microsoft DNA white paper , Microsoft web site 1997
[3] Microsoft transaction server white paper

（来稿时间：1998 年 3 月）

© 中国科学院软件研究所 http://www.c-s-a.org.cn