计算机科学2000Vol. 27№ 2

基于 MIDAS 构建多层分布式结构及应用

The Building and Application of Multilayer Distributed Structure Based on MIDAS

张 虹 甄青坡

（中国矿业大学计算机科学与技术系 徐州221008）

Abstract The multi-tiered distributed C/S system is superior to the two-tiered C/S system in many ways But it is very difficult to develop a multi-tiered distributed application system successfully. A good technique to develop it is described in this article, which is the MIDAS technique of Borland Corporation. The paper introduces and anaylses the structure, the technique and the operating process of building multi-tiered distributed application system based on MIDAS, and explains its superiority with examples.

Keywords MIDAS, Multi-tiered distributed, Client application program, Application server, Database server

1. 引言

随着计算机网络技术的迅速发展，基于 Client/Server(C/S)模型的分布式应用越来越广泛，追寻数据处理的发展史，自1980年第一个数据库管理系统的出现宣布了数据库世纪的开始，那时的观念是由应用程序控制数据库，这种数据处理的模式一般称为单层结构（1-Tier）。随着 LAN、PC 机的广泛应用，以及 RDBMS、RAD 技术的成熟，数据库应用开始转向 C/S 结构，即两层结构(2-Tier)。这种结构在近十年内不但得到了广泛的运用，而且还相当成功，与单层的结构相比，这种开放式体系结构的耗费大大降低，分布式的计算结构能充分利用整个系统的资源，客户端的 GUI 使用户操作更加方便，RAD 使开发人员能够快速地开发出各种应用，然而，在两层 C/S 结构成功的背后却逐渐暴露出其构架上的缺陷。其中最明显的问题表现在应用程序的伸缩性和维护等方面，为了克服两层 C/S 结构上的不足，开拓多种平台加入到 C/S 模型中，方便客户对数据处理的要求。于是，数据库管理系统又向着多层应用发展，即在传统的 C/S 结构中，增加了应用程序服务器。这种新的结构就是所谓的多层结构(n-Tier 或 Multi-Tier)。应用程序服务器包括了统一的界面、业务规则的制定和数据处理逻辑的规定等。多层应用服务技术允许分割应用程序，本地计算机上无须安装一整套数据库工具，就可以在另一台机器上存取数据。同时它允许对业务规则和进程进行集中管理，并在整个网络上分发、实现进程负载的动态调节。与两层 C/S 结构相比，三层应用具有：系统维护更加容易；对象可重用；更高的开发效率；安全性更高等诸多优点[3,7]。

众所周知，开发服务器级的应用程序要比开发单纯应用级的程序困难得多，有很多系统服务需要考虑。如果没有一种好的工具，对于大多数程序员来说，开发一个复杂的多层结构应用系统是难以实现的。为此，Borland 公司推出了在 Delphi 环境中独具特色的开发多层结构的技术和工具集—MIDAS。本文介绍并分析了 MIDAS 构建多层分布式应用系统的结构、技术及工作过程，并以实例说明基于 MIDAS 技术构建多层分布式应用系统及优越性。

2. MIDAS 技术

MIDAS（Mutli-tier Distributed Application Services Suite，多层分布式应用程序服务器）是基于 Borland 的分布式数据技术，至少包括两方面的内容：内置在 Delphi 组件中，用户很容易地使用 DCOM、Socket 或 OLEnterprise 连接两台机器，并在两者之间来回传输数据集；OLEnterprise 产品对分布式计算和负载平衡提供超强的支持，该工具提供 DCOM 的选择方法简化了连接两台机器的任务，尤其是对两台运行 Windows95/98的机器更是如此。OLEnterprise 使用户能

张虹 副教授，主要从事多媒体技术与软件工程的教学和科研工作。

访问 ObjectBroker，它允许在几个服务器中随机分配负载。运用 MIDAS 将极大地降低程序员开发多层结构的 C/S 应用的难度[1~6]。

2.1 MIDAS 构建多层结构

MIDAS 技术是构建多层分布式体系结构的关键。构建的结构模式可分为三层：第一层是数据库服务器，提供数据的存储和管理功能；第二层是应用服务器，集中管理业务规则和客户端与 RDBMS 的数据交换，也叫作数据代理；第三层是瘦客户机，由客户应用程序完成数据显示及用户界面的功能。典型的三层应用结构如图1所示，图中(a)为有 BDE 的三层应用结构，(b)为无 BDE 的三层应用结构。

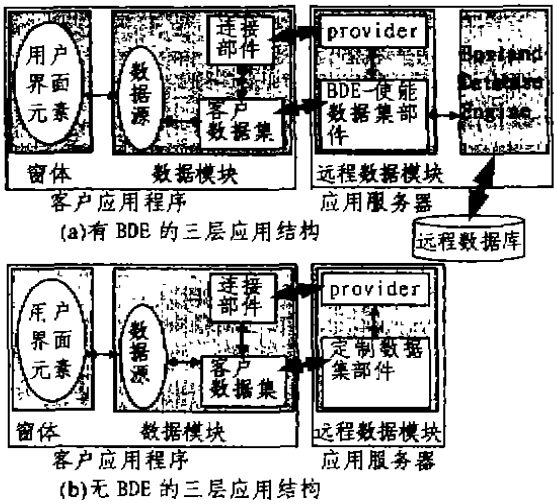

图1 典型的三层应用结构

客户应用程序(客户端层)包括窗体和数据模块，提供了一个可视化的接口，用来表示信息和收集数据，确保用户与应用程序之间紧密结合起来以处理某项业务。客户端层一般表现为用户界面，而且通常位于最终用户工作站上的一个可执行程序中。然而，有时候几个服务可能位于不同的分离的部件之中[5]。

应用程序服务器(中间层)的关键部件是远程数据模块，它是联系客户端和服务端的"桥梁"。它们响应用户(或其它的业务服务)发来的请求，并对相应的数据运用形式化的过程和业务规则去执行某种业务任务。如果所需要的数据驻留在数据库服务器上，通过它们可以获得所需的数据服务，或运用需要的业务规则。用户不能够直接与数据库打交道。

数据库服务器层提供数据服务，包括数据的定义、维护、访问和更新以及管理并响应业务服务的数据请求。

2.2 MIDAS 多层分布式技术

无论是应用服务器端还是客户端，MIDAS 技术需要有 DBCLIENT.DLL 的支持，这个动态链接库用于管理数据包。基于 MIDAS 的多层应用程序需要用到一些特殊的构件，这些构件分为四大种类：1)对象库中的远程数据模块。它与普通的数据模块有些相似，不同的是，远程数据模块可以作为 COM 服务器或 CORBA 服务器让客户程序访问它的接口；2)TDataSetProvider 和 TProvider 构件。这两个构件用在应用服务器端，主要作用是提供 IProvider 接口，客户程序通过 IProvider 接口获得数据和更新数据集；3)TClientDataSet 构件。是一个从 TDataset 继承下来的但不需要 BDE 的构件；4)MIDAS 连接构件。包括 TDCOMConnection、TSocketConnection、TCorbaConnection TOLEnterpriseConnection、TMIDASConnection 和 TRemoteServer。其中，TMIDASConnection 和 TRemoteServer 是为了兼容 Delphi3的代码而保留的。MIDAS 连接构件的作用是为客户程序定位服务器和 IProvider 接口。每个 MIDAS 连接构件都以一种特定的通讯协议工作[1,2,5]。

对于最终用户来说，多层体系结构中的客户程序与两层体系结构中的应用程序没有什么区别，多层体系结构中的客户程序是通过应用服务器提供的 IProvider 接口获得数据的，也可以通过 IProvider 接口申请更新数据。若使用 MTS 的时候，可以不使用 IProvider 接口。不使用 IProvider 接口的好处是，可以充分发挥 MTS 在处理事务方面的特长。在客户程序中，MIDAS 连接构件扮演着极其重要的角色。不同的 MIDAS 连接构件使用不同的通讯协议。

应用服务器的关键部件是远程数据模块，它提供了 IDataBroker 接口。当客户程序与应用服务器建立了连接，就通过 IDataBroker 接口来获得 IProvider 接口。Delphi 4 支持三种类型的远程数据模块：1)TremoteDataModule，这是一个支持双重接口的自动化服务器，这种类型的远程数据模块适合于使用 DCOM、TCP/IP 或 OLEnterprise 方式；2)TMTSDataModule，这也是一个支持双重接口的自动化服务器，用这种类型的远程数据模块创建的应用服务器是 Active Library，即动态链接库，适合于使用 DCOM、TCP/IP 或 OLEnterprise 方式；3)TcorbaDataModule，这是 CORBA 服务器，适用于与 CORBA 客户通讯。上述三种远程数据模块都可以作为容器，但只能放置非可视的构件。另外，远程数据模块上一般要放一个或几个 TDataSetProvider 或 TProvider 构件来提供 IProvider 接口。远程数据模块上也可以放 TDatabase 构件和 TSession 构件。

在客户程序与应用服务器之间，Delphi 4提供了四种不同类型的连接方式或者说通讯协议，包括 DCOM、TCP/IP、OLEnterprise 和 CORBA。这些不同的连接方式各有利弊，到底选择哪种连接方式，取决于客户的数量、客户的分布情况以及怎样发布应用程序。DCOM是一种最直接的连接方式，它不需要专门的运行期软件支持。不过，Windows 95不支持 DCOM，除非安装了DCOM95程序。要使用 MTS 安全服务，最好使用 DCOM 连接方式。MTS 的安全服务是基于角色的，当一个客户通过 DCOM 访问 MTS 时，DCOM 会告诉 MTS 有关客户的信息，MTS 据此来决定客户的角色。如果用其他连接方式，需要有专门的运行期软件支持，客户的调用首先被传递给这些运行期软件而不是 MTS，MTS 就不能尽快指派角色。TCP/IP 连接方式的适合范围非常广泛，例如，如果客户程序要以 ActiveForm 的形式分布在 Web 上，最好采用 TCP/IP 连接方式，因为您无法肯定下载 ActiveForm 的计算机是否支持 DCOM，而支持 TCP/IP 的环境是很普遍的。要使用 TCP/IP 连接方式，应用服务器端必须运行一个专门的运行期软件 ScktSrver.exe 或 ScktSrvc.exe，其中，ScktSrvc.exe 只适合于 Windows NT，可以作为一个服务在后台运行。与 DCOM 连接方式不同的是，客户的请求首先传递给 ScktSrver.exe 或 ScktSrvc.exe，然后再创建远程数据模块的实例，而不是由客户的调用直接创建远程数据模块的实例。客户程序上的 MIDAS 连接构件通过 IProvider 接口与 ScktSrver.exe 或 ScktSrvc.exe 通讯。不过，客户程序很有可能在没有正常释放对 IProvider 接口的引用之前出现异常，而 TCP/IP 连接方式无法检测到这种情况，更无法通知应用服务器，因此，有可能造成应用服务器上的资源被占用后得不到释放的后果。如果要在应用服务器端使用 Business Object Broker，就要使用 OLEnterprise 连接方式。此时，应用服务器端和客户端都要安装 OLEnterprise 运行期软件，基于 CORBA 的客户程序和应用服务器可以与其他基于 CORBA 的应用程序无缝对接[3]。要使用 CORBA 连接方式，需要 ORB 的支持，它提供了类似于 Business Object Broker 的功能。

2.3 多层 C/S 工作过程

用户首先要启动客户应用程序，客户应用程序将试图连接应用服务器，如果应用服务器还没有运行，客户应用程序将激活应用服务器，并从中获得 IProvider 接口。客户应用程序向应用服务器请求数据。如果 TClientDataSet 的 FetchOnDemand 属性设为 True，客户应用程序会根据需要自动检索附加的数据包如 BLOB 字段的值或嵌套表的内容。否则，客户应用程序需要显式地调用 GetNextPacket 才能获得这些附加的数据包。应用服务器收到客户应用程序的请求后，就从远程数据库服务器那儿检索数据，并打包返回给客户应用程序，客户应用程序收到数据包后把包打开，然后显示或进行处理。用户对数据进行编辑修改，然后向应用服务器申请更新数据，实际上也要打包。应用服务器收到客户应用程序的申请后，就向远程数据库服务器申请更新数据。如果出错，应用服务器就把出错的记录返回给客户应用程序去核对。客户应用程序核对或修改数据后，既可以放弃此次更新，也可以继续此次更新。

3. 基于 MIDAS 的应用

远程教育是一种不受地理空间限制的教育方式，教师与学生在不同的地区完成一系列教与学的行为，如课程的学习、讨论、考试等。远程教育包含所有程度的教育，从初等教育到高等教育，其活动可以是正规教育，也可以是非正规的自觉性教育。在计算机信息技术、网络技术和多媒体技术迅速发展和普及的今天，为了适应教育的发展，提高全民族的文化素质，我们开发了一个基于 MIDAS 技术构建的三层分布式网络考试系统。网络考试系统结构如图2所示。

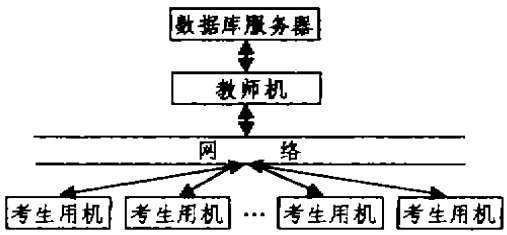

图2　网络考试系统结构

教师机位于应用程序服务器一层，考生用机则属于客户端一层。设计过程中，考生机（客户端）与教师机（应用程序服务器）之间的连接采用 Socket 连接方式，这样客户端几乎达到了"0"配置，教师机端运行 ScktSrver.exe 软件。

按照考试的流程，教师机主要做如下工作：选择考试试卷→接受学生登录请求→结束登录→设置考试时间→发放考试试卷→下达答卷指令→结束考试并回收答卷。

考生在登录完毕接到答卷指令后，在计算机上完成试题的浏览、答题以及交卷等过程。

结束语　利用 Borland MIDAS 技术我们可以非常方便地构建多层 Client/Server 分布式应用系统，它

不仅具有一般的多层 Client. Server 分布式应用系统的优点，而且 Borland 公司针对多层分布式应用的开发还提供了最完美的支持。例如，通过浏览器和 Web 服务器也能在客户端没有任何数据库工具的情况下观察远程机器上的数据集；通过分布式数据集可大大缩减网络通信量；提供访问数据库约束条件，当从服务器上卸载数据时，可以同时卸载一套自动执行的约束条件；Borland 多层计算的一个重要功能是将数据库的负载分散到多个服务器上；Remote DataBroker，其结构的精髓是让每一个客户端不再需要 BDE，取而代之的是一个中央化的 BDE，以集中管理的方式降低每一个客户在 BDE 上所需的开销和复杂度；Constraint Broker，它所扮演的角色是保证所有客户数据的一致性及数据的完整性；BusinessObjectBroker，其目的是给一些关键性的商业应用程序提供一个快速且可信赖的使用环境，为了满足这种高层次的要求，Business Object-Broker 会自动地将应用程序做适当的划分，并复制重要的业务规则到每一个区间，以达到速度的要求。

参 考 文 献

1 Microsoft DCOM Technical Overview. Available at: http://www.microsoft.com

2 Building Scalable Application with Windows NT and COM. Available at: http://www.borland.com/midas/dcom

3 徐新华. Delphi 4核心编程技术. 希望电脑公司出品，1998.11

4 Developer's Guide-Borland Delphi4 for Windows95 and WindowsNT. INPRISE Coporation, 100 Enterprise Way Scotts, Valley, CA 95066-3249

5 Borland 的 MIDAS 技术. Available at: http://bbs.tsinghua.edu.cn

6 徐新华，Delphi 3编程指南. 宇航出版社，1998.6

7 鲍泓，佟建新，尉林明，等. Windows NT 组网技术. 电子工业出版社，1998.4

（上接第96页）

经过上述修改后的算法能够保证不同顶对之间互不干扰，从而避免了上述情况的发生。

上述算法的时间复杂度为 $\sum_{(a,b)\in E}O(nmR_{ab})=O(nmR_{\sum})$，由此得到以下定理：

定理2 给定无向连通单图 $G=(V,E)$，可在 $O(nmR_{\sum})$ 时间内枚举 G 的所有极小顶割，其中 $n=|V|$，$m=|E|$，$R_{\sum}=\sum_{1\le i\le n,(v_i,v_j)\in E}R_{v_iv_j}$。

对于连通单图 G，有 $m<n^2$，当 G 为稀疏图时，$m=O(n)$，此时 $O(nmR_{ab})=O(n^2R_{ab})$。显然我们的第一个算法改进了 *Kloks* 的结果 $O(n^3R_{ab})$。又因为 $R\le R_{\sum}\le n^2R$，$O(nmR)\le O(nmR_{\sum})\le O(n^2mR)$，所以我们的第二个算法也改进了 *Kloks* 的结果 $O(n^5R)$。

3. 推广

给定 G 中两个不相邻的顶子集 A，B，若要求所有极小 (A,B) 顶割，可以先将顶子集 A 和 B 退化成两个顶 a 和 b，对于 $V-A-B$ 中的顶，若与 A（或 B）中某顶相邻，则在 a（或 b）和该顶之间连一条边，从而将图 G 变成图 $G'=(V',E')$，其中：

$$V'=(V-A-B)\cup\{a,b\}$$

$$E'=E(V-A-B)\cup\{(a,v)\mid v\in V-A-B,\text{且}\exists\ u\in A,\text{使}(u,v)\in E\}\cup\{(u,b)\mid u\in V-A-B,\text{且}\exists\ v\in B,\text{使}(u,v)\in E\}$$

然后在 G' 上使用上一节的算法 separator，求出 G' 中的所有极小 (a,b) 顶割，而这些极小 (a,b) 顶割就是 G 中的所有极小 (A,B) 顶割，所以通过上面的过程可以求得 G 中的所有极小 (A,B) 顶割。

现分析上述过程的时间复杂度如下：将 G 变成 G' 的过程中，计算 V' 所花时间为 $O(n)$，计算 E' 所花时间为 $O(m)$；由于 $|V'|=n-n_A-n_B+2$，$|E'|\le m$，所以在 G' 上求所有极小 (a,b) 顶割的时间为 $O(m(n-n_A-n_B)R_{AB})$。故总时间为 $O(m(n-n_A-n_B)R_{AB})$，于是得到以下定理：

定理3 给定无向连通单图 $G=(V,E)$，$A\subset V$，$B\subset V$，且 $A\cap B=\varnothing$，A 与 B 不相邻，则可在 $O(m(n-n_A-n_B)R_{AB})$ 的时间内枚举 G 的所有极小 (A,B) 顶割，其中 $n=|V|$，$m=|E|$，$n_A=|A|$，$n_B=|B|$，R_{AB} 为所有极小 (A,B) 顶割的数目。

最后，一个极富挑战性的问题是能否在 $O(n^2R_{ab})$ 时间内枚举所有极小 (a,b) 顶割，这也是我们下一步的工作。

参 考 文 献

1 Ariyoshi H. Cut-set graph and systematic generation of separating sets. IEEE Trans. Circuit Theory, 1972, CT-19: 233～240

2 Goldberg L. A. Efficient algorithms for listing combinatorial structures. Cambridge Univ. Press, Cambridge, 1993

3 Kanevsky A. On the number of minimum size separating vertex sets in a graph and how to find all of them. In: Proc. 1st Ann. ACM-SIAM Symp. Discrete Algorithms, 1990. 411～421

4 Arnberg S. Efficient algorithm for combinatorial problems on graphs with bounded decomposability a survey. BIT, 1985, 25: 2～23

5 Kloks T, Kratsch D. Listing all minimal separators of a graph. SIAM J. Comput., 1998, 27(3): 605～613