基于 Delphi 的 MIDAS 分布式体系结构分析与实现

Analysis and Implementation of MIDAS Distributed Architecture Based on Delphi

苏志芳　陈和平　(武汉科技大学信息科学与工程学院 430081)

摘要:本文首先介绍了基于 Delphi 的 MIDAS 多层分布式数据库技术的基本特点及其发展历程,然后阐述了 MIDAS 三层分布式体系结构工作流程及典型构建过程,最后结合开发实例给出了 MIDAS 的具体实现方法并对执行效率问题进行了讨论。

关键词:MIDAS　多层分布式　Delphi　无状态对象

1　引言

MIDAS 是 Multi – tier Distributed Application Services Suite(多层分布式应用程序服务包)的缩写,在 Delphi(或 C + + Builder)企业版里被用来创建多层应用程序,是 Inprise 公司设计的用于开发多层应用系统的透明数据库中介引擎。

在 Delphi 6、7 及以后的版本中,MIDAS(更名为 DataSnap)激活了 Web 服务功能,强化了原有的功能,如:不必用底层的 COM 界面或是复杂的 API 函数就可以轻易地处理 XML/XSL 文档和报表;MIDAS 更直接以 XML 形式组织数据包等。因此,基于 MIDAS 的多层分布式结构为实现 Internet/Intranet 和电子商务等应用系统提供了较佳的解决方案。

2　MIDAS

多层分布式开发模型解决了 C/S 模式的维护成本高、客户端臃肿等弊端,使得系统的稳定性、执行效率、容错能力和负载平衡能力得以全面提高。

2.1　三层体系结构

三层体系结构指逻辑上的三层,即应用表示层、应用逻辑层和数据层。应用表示层,在系统模型中又名客户端,主要负责用户端界面,提供给用户一个操作方便且简单快捷的应用服务接口;应用逻辑层(或为应用服务器),是整个结构中最重要的部分,实现应用程序的应用逻辑处理;数据层(又为数据库服务器),则负责数据的存取和管理。应用逻辑层将业务规则、数据访问及合法性检验等工作放到了中间层进行处理。通常情况下,客户端不直接与数据库进行交互,而是通过通信协议与中间层建立连接,再经由中间层与数据库进行交互。

分布式中间件 MIDAS 等的使用使得多层分布式数据库应用系统的开发工作变得简单易行。

2.2　MIDAS 体系结构模型及数据处理流程

Delphi 对多层分布式应用程序的支持主要得益于其 MIDAS 技术,该技术允许分割数据库应用程序,并实现对商业规则和进程的集中管理。其典型的三层结构如图 1 所示。

2.2.1　客户端

从逻辑结构上看,客户端主要由三部分组成:

(1) 客户端的 DataModule 中包含与应用服务器建立连接的 RemoteServer 组件,客户端通过 RemoteServer 组件与应用服务器的 IAppserver 接口连接,以此进一步连接 DataSetProvider 接口,从而通过 DataAccess 组件实现数据的获取和更新操作。

(2) ClientDataSet 组件支持数据的存取、编辑、浏览、约束和过滤等功能。

(3) 在客户端的 Form 中,由数据感知组件(如 DBGrid、DBEdit、DBNavigator 等)形成与用户交互的接口。

2.2.2　应用服务器

作为中间层的应用服务器是系统的核心,也是连接客户端和远程数据库的重要桥梁。该层主要由远程数据模块构成。

(1) 远程数据模块 Remote Data Module(RDM)是一个支持双重接口的自动化服务器,它自身提供了

IDataBroker(IAppServer)接口。客户端利用该接口与数据供应接口 DataSetProvider 通信。RDM 是一个容器,可容纳访问远程数据库服务器的 DataAccess 对象和充当数据代理的 DataSetProvider 对象等。

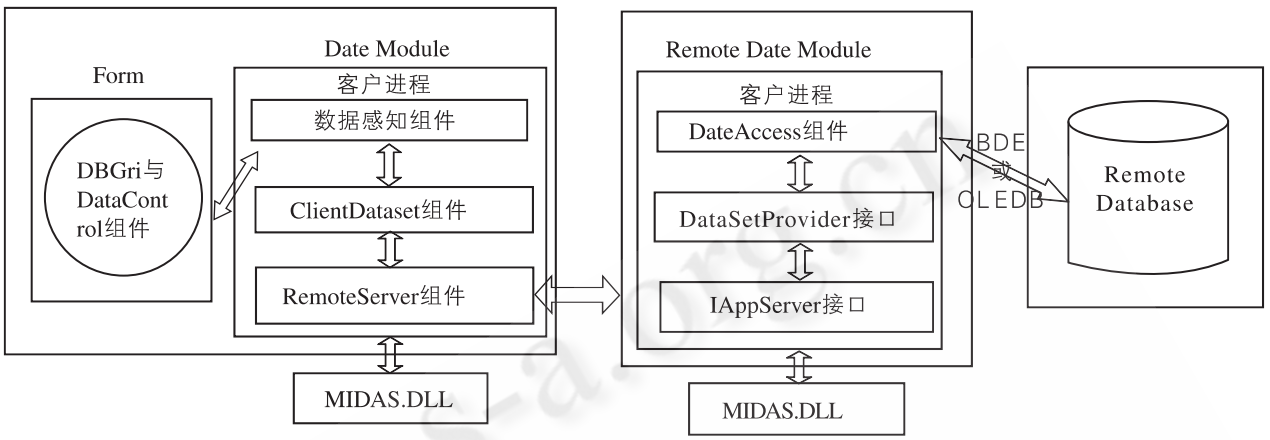

图 1 MIDAS 体系结构模型图

(2) 数据集组件 DataAccess 主要包括 Table,Query,StoreProc 等,其主要功能是实现远程数据库的访问,该组件通过 BDE/ADO/OLE DB 等引擎接口访问远程数据库。

(3) 数据供应组件 DatasetProvider 充当 DataSet 对象与客户端 ClientDataSet 对象间的数据代理。该组件有两大功能:

① 通过 DataSet 对象从数据库服务器获取数据并封装成数据包后回传给客户端的 ClientDataset 对象;

② 将客户端 ClientDataset 对象中要求更新的数据提交给数据库服务器,并把因各种原因不能实现更新的数据存入日志后回传给客户端。

(4) MIDAS.DLL 用于管理与组织应用服务器提供者组件和客户端数据集组件上的数据包。应用服务器上的数据包经由 DCOM 或 Web(HTTP)通信协议到达客户端。

2.2.3 数据库服务器

后台数据库 Oracle 或 Informix 等是独立的,存放着用户的所有业务数据,通过 BDE、OLEDB 或 ODBC 等数据访问接口和应用服务器进行通信。

2.2.4 MIDAS 数据处理流程

在上述模式下,系统数据处理流程如下:

(1) 客户端提交请求,通过数据感知控件(DBGrid 等)通过 DataSource 对象连接到 ClientDataSet 组件;

(2) 客户端 MIDAS.DLL 封装 ClientDataSet 组件的数据请求,通过 RemoteServer 组件连接到远程的应用程序服务器;

(3) 该数据封包通过连接通道(TDCOMConnection 组件或 WebConnection 组件)找到应用服务器;

(4) 应用服务器的 MIDAS.DLL 对数据解包后,通过 IAppServer 接口获得用户请求;

(5) DataSetProvider 组件与 IAppServer 接口通信,再与数据集组件 DataAccess 建立连接;

(6) 数据集组件通过数据库引擎(OLEDB 或 BDE)访问远程数据库,并获得结果数据;

(7) 按照上述用户请求传递路径逆向将最终结果返回给客户端。

2.3 实现步骤

基于 MIDAS 的三层分布式应用的主要实现步骤分为应用服务器构建和客户端构建两部分。

(1) 构建应用服务器

① 生成一个应用程序工程,并在其中建立一个 RDM;

② 在 RDM 中添加所需的 ADOQuery 组件和 DataSetProvider 组件,设置 ADOQuery 的连接属性 ConnectionString 和远程数据库建立连接,并设定 DataSetProvider 的 DataSet 属性,使其与某个 ADOQuery 数据集组件建立连接;

③ 编写应用逻辑的程序代码,实现业务规则;

④ 编译、运行该程序,即可实现应用服务器注册。

因程序是 Web 服务程序,因此必须与 Web 服务器一起注册,还必须发布描述程序可调用接口的 WSDL 文档。

(2) 构建客户端

① 添加一个应用工程,设置一个 ActiveForm 窗体(用于浏览器调用)或一般的 Form 窗体;

② 在 ActiveForm 窗体上放置所需要的 SimpleObjectBroker 组件(为客户端提供简单的负载平衡和容错能力,可动态地选择应用服务器)、WebConnection 组件(可通过 MIDAS 的 HTTP 协议实现客户端与应用服务器之间的通信)和 ClientDataSet 组件。在 SimpleObjectBroker 的 Server 属性中添加应用服务器的 IP 地址,使 WebConnection 的 ObjectBroker 属性和 SimpleObjectBroker 建立连接,为 ClientDataSet 的 RemoteServer 属性指定一个 MIDAS 连接组件并设置 ProviderName 属性为应用服务器上的 DatasetProvider 组件名。设置完毕后,客户程序就可以通过 Remoteserver 组件与应用服务器通信了;

③ 放置 TMidasPageProducer(InternetExpress)组件,编写程序将所需数据信息显示在该组件指定的 ActiveForm 上。

至此,一个基本的 MIDAS 分布式体系构建完毕。

3 MIDAS 分布式体系中的效率问题

在分布式系统中,系统的效率受到各层执行效率和网络带宽等诸多因素的影响。因此,如何提高执行效率已经成为开发信息系统的一个重点。MIDAS 提供的应用服务器的 RDM 对象可重用技术(如无状态对象技术)就是其中的解决方法之一,它能有效利用应用服务器资源,增加同时可服务的客户端数量。

无状态对象(Stateless Object)是指对象在提供了调用者要求的服务之后,该对象的内部状态完全消失,不为上次的调用保留任何的内部状态信息。当应用服务器为有状态对象(State Object)时,客户端与应用服务器端相对应的某个 RDM 之间需要始终保持连接状态。假设应用服务器中的 TDataSetProvider 组件为有状态对象,TClientDataSet 以自动分段方式从应用服务器的 TDataSetProvider 组件读取数据,那么在客户 C1 获取全部数据之前,其他客户将不能申请使用客户 C1 所用的应用程序服务器。此时无法实现 RDM 重用,即需要为每一个此类连接新建一个 RDM 实例。解决此类问题的方法是将应用服务器中的 TDataSetProvider 组件设计为无状态对象(即将 TClientDataSet 对象的 FetchOnDemand 属性设定为 false,则它所连接的 TDataSetProvider 对象就自动成为无状态对象)。按客户的要求服务完后,应用服务器的内部状态将完全消失。使得应用服务器实例能服务于多个客户,从而有利于提高应用系统的执行效率、可扩展性和可重用性。

在笔者设计的某电器设备计价管理系统中,各个用户可能通过各自的客户端同时浏览和查询数据库服务器中的电器元器件信息。由于元器件的数量很多,客户端每页只能显示其中的 15 条信息。各客户都可能在每一页信息上停留一段时间,然后再进入下一页。因此,每个客户都需要维持当前记录的状态信息。

如果不采用无状态对象,则应用服务器必须为每个连接建立一个 RDM 对象并一直维持到客户端应用程序结束。当多个用户同时访问数据库时,就可能出现应用服务器负载过重。该应用服务器的 RDM 被设置为无状态对象,其中加入一个 Table 对象 CpmlTable,与数据库服务器中的产品目录表连接,再加入 TDataSetProvider 对象 Cpmlprovider,并和 CpmlTable 表建立连接。当应用服务器收到客户端的获取下一个数据包的请求后,先触发 Cpmlprovider 对象的 BeforeGetRecords 事件,然后从 CpmlTable 表获取一个含 15 条记录的数据包,经由 Cpmlprovider 数据代理对象回传给客户端。具体代码略。

4 结束语

MIDAS 技术结合了分布式计算模式和组件构建技术的优点,为开发分布式信息系统提供了解决方案。本文针对 Borland 公司的 MIDAS 多层分布式体系结构的主要特点和相关技术进行了分析研究,并具体给出了构建一个 MIDAS 应用系统的实现步骤。

参考文献

1 李维著,Delphi 5.X 分布式应用系统篇[M],机械工业出版社,2000,4-26。

2 新智工作室编著,Delphi 7 程序设计与实例[M],电子工业出版社,2003,430-446。

3 祝建中,无状态对象及应用,杭州师范学院学报[J],2003,07(4):26-29。