MIDAS技术的实现与可靠性分析

龙 艳
（武汉理工大学 计算机科学与技术学院，湖北 武汉 430070）

摘 要：MIDAS为开发多层分布式应用系统提供的一个中间透明引擎，能有效地利用DCOM、TCP/IP等技术在Internet/Intranet上建立结构为“瘦客户端+应用程序服务器+数据库系统”的多层分布式应用程序，在此结构下可利用对象代理实现分布式负载均衡与容错，从而提高系统的性能。

关键词：MIDAS；应用服务器；负载平衡；容错
中图分类号：TP311.52 **文献标识码**：A

The realization and reliability analysis of MIDAS technology

LONG Yan
(Department of Computer Science and Technology, Wuhan University of Technology, Wuhan 430070, China)

Abstract: MIDAS, provided by Delphi, is a transparent medium to implement mutli-tier distributed application system. It is used to build the structure of “Thin Client + Application Server + Database System” in Internet/Intranet through the technology of DCOM, TCP/IP etc. In this architecture, object broker can be used to implement distributed-load balancing and fault tolerance. Thus the performance of system is improved.

Key words: MIDAS; application server; distributed-load balancing; fault tolerance

Internet/Intranet 的兴起使得企业很方便地使用MIS（企业管理系统）来进行规划管理。目前大多数MIS采用两层C/S结构，但随着使用的深入，该结构逐渐暴露出其架构上的缺陷，如应用程序的可伸缩性和维护，同时，如何控制数据的统一性和完整性也成为问题。

新一代的数据库管理系统在传统的C/S结构中增加了应用服务器，这种新的结构就是所谓的三层体系结构。三层体系结构指逻辑上的三层，包括应用表示层、应用逻辑层和数据层。应用表示层又名客户端，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，负责数据的存取和管理。应用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功能。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。多层应用服务技术允许分割应用程序，这样，本地计算机无需安装一整套数据库工具，便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机种管理，并在整个网络上分发，实现进程负载的动态调节。

1 MIDAS技术

MIDAS的全称是Multi- tier Distributed Application Services Suite(多层分布式应用程序服务包)。Delphi中所有强大又奇妙的分布式多层能力都来自MIDAS的功能。MIDAS是Delphi多层应用系统的技术核心，是Delphi用来开发多层应用系统所使用的中介透明引擎。通过MIDAS，程序员可以使用相同的组件存取不同的后端应用程序服务器。此外MIDAS也提供了容错能力、负载平衡能力以及高执行效率的能力。

多层结构其实是对传统C/S结构的扩展，是把一个数据库应用程序分解成几个逻辑部分，客户端程序集中处理数据显示部分和用户与数据之间的交互。其优点可归纳为：

(1)将数据处理及通信功能封装在一个共享的中间层里。不同的客户端程序都能访问这个中间层，这样就

避免了为每个客户程序复制数据处理部分而产生的冗余。

(2)缩小了客户端程序的规模，使得客户端程序更容易进行开发。这是因为不需要安装、配置和维护数据库连接软件。

(3)采用分布式数据处理过程。将一个应用程序要处理的任务分在几台机器上进行处理，从而提高了程序执行的性能。

(4)提高了数据的安全性。将不同的数据功能封装成一定的中间层，通过提供安全控制结合容错技术，可以开发出更安全的应用系统。

2 MIDAS 技术架构和工作原理

MIDAS 技术实现的架构图如图 1 所示

2.1 客户端

从逻辑结构上看，客户端主要由三部分组成。首先，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。再者 ClientDataSet 组件支持数据的存取、编辑、浏览、约束和过滤等功能。最后在客户端 Form 中，由数据感知组件如 DBGrid、DBEdi、DBNavigator 等形成与用户交互的接口。

2.2 应用服务器

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。

远程数据模块 RemoteDataModule 是一个支持双重接口的自动化服务器，它自身提供了 IAppServer 接口。客户端利用该接口与数据供应接口 DataSetprovider 通信。远程数据模块是一个容器，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。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。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，并把由于各种原因而不能实现更新的数据存入日志后回传给客户端。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。

2.3 数据库服务器

后台远程数据库存放着用户的所有业务数据，通过 BDE、OLE DB 或 ODBC 等数据访问接口和应用服务器进行通信。

在一个基于 MIDAS 的三层应用中，客户端通过应用服务器得到数据和进行数据更新的过程，通常按照如下顺序和方式实现：

(1)用户启动客户端应用。客户端连接到应用服务器，如果应用服务器尚未运行，它将被启动。客户端从应用服务器得到一个 IAPPServer 接口。

(2)客户端从应用服务器请求数据。

(3)应用服务器获取数据，为客户端打包数据，返回一个数据包给客户端。

(4)客户端解开数据包，显示数据给用户。

(5)当用户与客户端程序交互时，数据被更新。这些改动被客户端记录在一个变更日志中。

(6)最后，客户端向应用服务器提请更新(applyupdates)，通常是响应于用户的一个操作。

(7)应用服务器解开数据包，并向数据库服务器递交更新(postupdates) 。

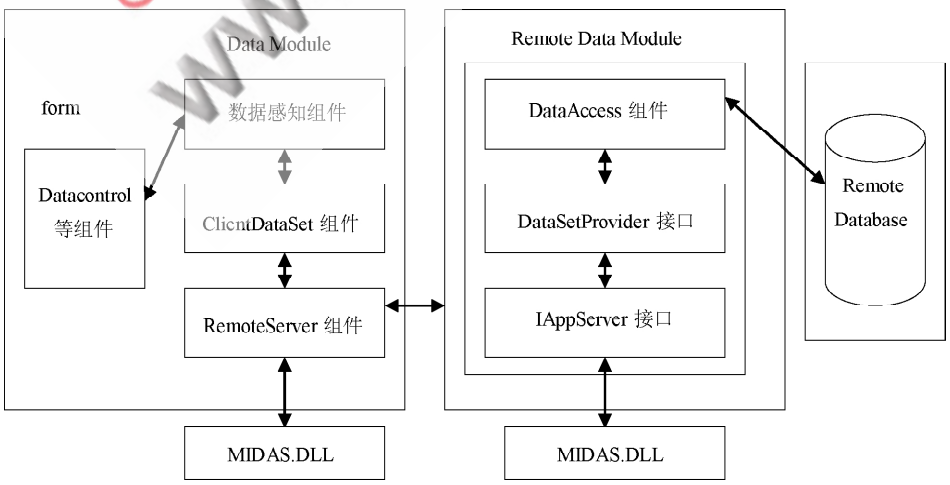

图 1 MIDAS 技术实现的架构图

(8)当应用服务器完成解析过程,它返回任何没有递交的记录给客户端,以供后来的进一步解析。

(9)客户端调整没有解析的记录。

(10)客户端从服务器得到数据刷新自己。

3 MIDAS 的实现

要创建一个多层 C/S 应用程序,首先要创建应用服务器,然后注册或安装应用服务器,只有应用服务器已注册并且正在运行的情况下,才能创建客户程序。对于客户程序来说,既可以在设计期间连接应用服务器,也可以在运行期间连接应用服务器。创建 MIDAS 应用服务器步骤如下:

(1) 创建一个远程数据模块(Remote DataModule),它允许客户以 DCOM、TCP 等方式访问此服务器;

(2)把一个数据集构件, 如 TTable、TQuery 或 TStoredProc 放到远程数据模块上,使它们能访问远程的后台数据库;

(3)把 TDataSetProvider 构件放在远程数据模块上,设置它的 DataSet 属性指定为要访问的数据库,其实就是第(2)步所放的数据集构件;

(4)编写代码,实现规则;

(5)保存、编译、注册或者安装应用服务器。

在多层体系结构中,一个客户程序至少需要一个 TClientDataSet 构件,作用是引入数据集。创建一个客户程序的步骤如下:

(1)新建一个项目,在项目中添加一个数据模块(DataModule);

(2)根据通信协议把一个或者几个 MIDAS 连接构件如 TDCOMConnection 加入到数据模块上,设置相关属性,指定和连接应用服务器,这与具体的 MIDAS 连接构件有关;

(3)把一个或几个 TClientDataSet 构件放到数据模块上,设置 RemoteServer 属性指定为第(2)步添加的 MIDAS 连接构件,设置 Providername 属性为应用服务器上的 TDataSetProvider 构件,这样客户程序就可以和应用服务器通信;

(4)把一个 TDataSource 构件放到数据模块上,设置它的 DataSet 属性为第(3)步添加的 TClientDataSet 构件,再把一个数据感知控件如 TDBGrid 放到窗体上,设置它的 DataSource 属性为 TDataSource 构件,至此,一个简单的客户程序创建完毕。

4 可靠性分析

在多层结构环境中,所开发的应用系统除了必须能够正确而且有效率地运行之外,如何让其更安全可靠,不会因为应用服务器或是数据库服务器故障而导致整个系统无法继续运行也是非常重要的。分布式多层应用系统的可靠性具体可以通过以下能力来体现:

(1)负载平衡能力。指能够根据应用程序服务器的不同负荷,动态分配客户端的连接,不至于有的应用服务器负载过重,而有的应用程序服务器负载过轻,使得所有的应用程序服务器的负载达到一个平衡。

(2)容错能力。指某台应用服务器发生故障时,原先连接到该服务器的客户端可以立刻连接到其他提供相同服务的应用服务器,并继续取得服务。

(3)暂存数据能力。指当所有的应用服务器都发生了故障,或是数据库服务器发生了故障,客户端应用程序应该有一种机制能够把用户更新的数据暂存在客户端机器中,等应用程序服务器或是数据库服务器恢复正常后,客户端应用程序可以把数据加载到系统中,再实际更新回数据库服务器中。

Delphi 在客户端提供了 TSimpleObjectBroker 组件来实现基本的负载平衡能力和容错能力,该组件实现了基于伪随机数算法的简单对象代理。在客户端的 Data Module 中加入一个 TSimpleObjectBroker 组件,并将其 LoadBalanced 属性设置为 true,然后把 Data Module 中各个 RemoteServer 组件的 ObjectBroker 的属性设置成该 TSimpleObjectBroker 组件,就能够实现伪随机意义下的负载均衡。Servers 属性存储一张能够执行应用服务器列表,该表的第 i 个表项 Items[i]由 ComputerName、Port、Enabled、HasFailed(应用服务器所在的计算机名、端口号、当前是否可用标记、连接失败标记)4 个域构成,并提供其中的机器名称给 TDCOMConnection 或是 TSocketConnection 作为连接的远程机器的名称。当 TDCOMConnection 或是 TSocketConnection 连接的机器发生故障时,TDCOMConnection 或是 TSocketConnection 可以从 TSimpleObjectBroker 取得一个新的能够执行应用程序服务器的远程机器名称,然后再连接到这台新机器以取得应用程序服务器的服务。为了实现暂存数据能力,TClientDataSet 提供了 SaveToFile 和 LoadFromFile 两个方法。当所有的应用程序服务器都发生了故障,或是数据库服务器发生了故障时,调用 SaveToFile 方法把 TClientDataSet 中所有的数据包括在客户端更新的数据保存到一个文件中,然后在应用程序服务器或是数据库服务器恢复正常后再执行客户端应用程序,调用 LoadFromFile 方法加载先前存储的数据到 TClientDataSet 中,最后再调用 ApplyUpdates 方法把客户端更新的数据更新回数据库中。

MIDAS 技术与 C/S 技术相比,尤其是在大中型数据库应用系统中,系统的稳定性、延展性和执行效率有很大提高。开发一个高效可靠的分布式多层数据库应用

(下转第 15 页)

相结合使用的事例较多，用法基本相同，只是功能有所差别。以用户登录模块为例，其主要代码如下：

用户名：<input type="text" id="username" name=" user. userName" />

密码：<input id="password" name="user. password" type="password" />

XWork.xml 的配置。从 JSP 页面上将用户输入的用户名和密码提交后，将转入配置文件 XWork.xml 中，通过这个配置文件寻找 Java 后台处理登录过程逻辑的类，同时将提交的内容转移到 Java 逻辑类中进行判断处理。

逻辑处理。与上文中 XWrok.xml 中的配置相对应，在 LoginAction 类中，使用 UserLogin 逻辑处理方法，判断用户名和密码是否与数据库中的数据相符，以此做出不同处理，并回显在页面上。UserLogin 方法实现主要代码如下：Userlogin：User user = lb.getLoginUser(username);session. setAttribute ("user", user);

另外，在数据库的导入过程中，根据数据库中表的名称，利用 Hibernate 框架将对象类 User.java 和操作类 UserDAO.java 导入到项目中，在处理逻辑的过程中直接调用，以此加速项目开发的周期。

前台页面响应。在后台根据用户名和密码做出判断，在前台显示结果，主要表现为：一是页面的转向，通过 XWork.xml 的配置得出要跳转到哪个页面。如上文程序所示，如果返回 Action. ERROR，则转向 error.jsp；如果返回 Action. SUCCESS，则转向 index.jsp。另一个是信息的反馈，通过 Session 或 Request 将需要传送到前台页面的值放到缓冲区里，然后显示在 JSP 页面相应的位置。

从上面这个登录流程的简单例子可以看出，在整个集成框架中不存在与业务逻辑无关的垃圾代码，也没有 SQL 查询语句，整个开发过程完全是面向对象的操作方式和可动态配置、可移植的 Xml 文件，大大提高了开发效率。

3.2 高校课程改革系统架构

在高校课程改革管理系统的开发中，通过使用基于 WebWork、Spring 和 Hibernate 的 J2EE 轻量级集成框架，加载开源框架类，依据 MyEclipse 中的配置，打开数据库连接，导入数据表的对象类和一些基本实现方法。特别是将存有高校专业课程的 Excel 表格中的数据较完整的导入数据库中（本文使用 SQL server 2000 数据库），以避免人为操作的失误，同时也提高了工作效率，为以后对数据库中存储的“专业课程”数据进行预处理做好铺垫，提高数据挖掘的准确性和真实性。课程改革系统架构如图 3 所示。

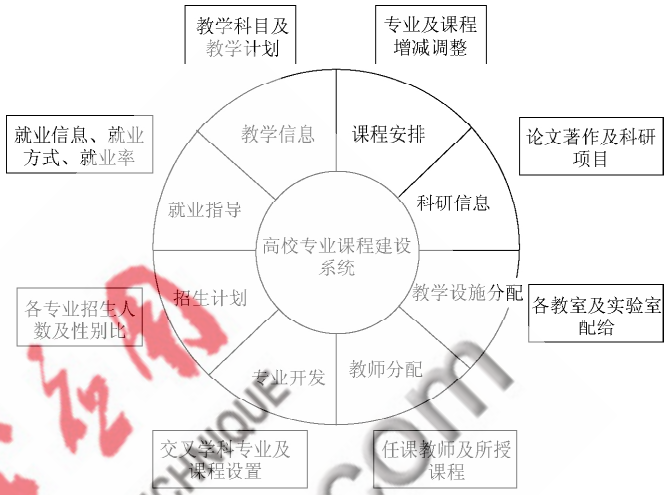

图 3 高校课程改革系统架构

综上所述，将 WebWork、Spring 和 Hibernate 三种开源框架集成在一起，形成一种轻量级 Web 开发架构，该架构充分发挥三者的优点，层次清晰，具有较高的伸缩性、可扩展性和可复用性，开发简洁、维护方便。通过对 J2EE 集成开源架构的分析和研究，将其应用在高校专业课程改革管理系统的开发过程中，使 Web 项目的设计和开发中具有一定的优势和广泛的应用前景。

参考文献

[1] LIGHTBODY P. WebWork in Action[Z].Manning Publications Co, 2005.

[2] RAIBLE M. Spring Live[Z].Source Beat Publishing,2004.

[3] 阎宏. Java与模式[M].北京：电子工业出版社,2005.

[4] JOHNSON R. J2EE设计开发编程指南[M].北京:电子工业出版社,2003.

[5] ECKEL B. Java编程思想[M].侯捷,译.北京: 机械工业出版社, 2002.

（收稿日期：2009-01-15）

（上接第12页）

系统，还必须考虑负载平衡、容错控制等能力，使用 MIDAS 的 ObjectBroker 技术可以实现并有效地提升系统的效率和健壮性。

参考文献

[1] 李维. Delphi 5.x 分布式多层应用系统篇[M]. 北京: 机械工业出版社，2000.

[2] TEIXEIRA S,PACHECO X. Delphi开发人员指南[M]. 北京: 机械工业出版社，2003.

[3] 王志刚. 基于 MIDAS 分布式多层系统的容错技术. 湖南师范大学自然科学学报, 2001, 24 (2): 15.

[4] 徐新华.Delhi高级编程——Database与MIDAS编程[M].北京: 人民邮电出版社，2000.

[5] 于重重. 基于三层ClientServer结构的管理信息系统的实现. 北京工商大学学报，2000，17（7）：35-38.

（收稿日期：2008-12-31）